

Resumé

Denne rapport omhandler udviklingsforløbet af et program, til at konvertere målene fra britiske til metriske i kageopskrifter. Der er specificeret krav og design, implementeret kildekode, og programmet er testet med unit-test, blackbox og whitebox test, og resultatet lever op til specifikationen.

Indhold

1	Indledning	4
2	Kravspecifikation	5
2.1	Planlægning	5
2.2	Risikovurdering	6
3	Design	8
3.1	Tekniske design beslutninger	9
4	Implementering	10
4.1	Measurement	10
4.2	MeasurementList	11
4.3	RecipeConverter	12
4.4	Ændringer igennem udviklingsforløbet	13
4.5	Indlæsning og skrivning af filer	13
4.6	Brugergrænseflade	14
5	Afprøvning	17
5.1	Test af ekstra opskrifter	18
5.2	Grænsetilfælde	18
5.3	blackboxtest	19
5.4	unit-test	20
5.5	whiteboxtest	21
5.6	Accepttest	24
6	Brugervejledning	26
7	Konklusion	27
A	UML diagrammer	30
B	Tidsplan	31
C	Møde referater	33

<i>INDHOLD</i>	3
D Test opskrifter	37
D.1 Lord Baltimore Cake	37
D.2 Maraschino Cherry Nut Cake	37
D.3 Large White Birthday Cake	38
D.4 Dirt Cake 1	38
E Kildekode	40
E.1 MainGUI.java	40
E.2 MeasurementGUI.java	47
E.3 FileIO.java	53
E.4 Measurement.java	55
E.5 MeasurementList.java	58
E.6 RecipeConverter.java	61
E.7 FileIOTest.java	65
E.8 MeasurementTest.java	66
E.9 RecipeConverterTest.java	69
E.10 Case1Test.java	73
E.11 Case2Test.java	80
E.12 Case3Test.java	88
F Testskemaer Blackbox og whitebox	92

Kapitel 1

Indledning

Rapporten her er blevet til i faget NIS 02 - projektstyring og evaluering i april/maj 2005, og er et samarbejde mellem Jogeir Anderssen, Anders Brystning, Jens Kristian Kræmmer Nielsen og Jacob Aae Mikkelsen. Den fungerer som eksamen i faget.

Rapporten dokumenterer et udviklingsforløbet, og er ikke ment som en brugerhåndbog, derfor henvender den sig mere til softwareudviklere, end til programmets brugere. Programmet, der er blevet udviklet, kan bruges til at konvertere Britiske mål til metriske, specielt med henblik på bageopskrifter, og denne rapport dokumenterer processen heraf.

Der er i kravspecifikationsafsnittet beskrevet de krav, der er blevet stillet direkte og indirekte i opgaveformuleringen. Der er udarbejdet en risiko vurdering for projektets gennemførelse. I design afsnittet er kravene uddybet, og de beslutninger, som er truffet for at få programmet til at virke, er beskrevet. Der er en gennemgang af de vigtigste funktionaliteter i programmet i implementeringsafsnittet, samt en beskrivelse af det grafiske layout i brugergrænsefladen.

Testning af det udviklede program fylder en stor del af projektet, idet der er gennemført både unit-test, blackbox, whitebox og bruger-accepttest, og de fejl der i løbet af udviklingsprocessen blev fundet er rettet og testet igen.

Selve programmet er af en beskeden størrelse, men funktionaliteten er god. Det virker efter hensigten, og lever op til målsætningerne, der er beskrevet i opgaven.

Kapitel 2

Kravspecifikation

Opgaven består i at lave et program, der kan omsætte måleenhederne i Britiske kageopskrifter (imperial measures(eng.)) til Danske måleenheder (metriske).

Ifølge opgaveformuleringen skal programmet implementeres i Java. Som input skal det tage vilkårligt valgte opskrifter med Britiske måle, og output skal være de samme opskrifter, men nu med målene omsat til metriske mål. Desuden skal der kunne tilføjes nye mål, og de allerede eksisterende skal kunne ændres.

På denne baggrund ser vi et program, der henvender sig til en person, der ikke nødvendigvis har nogen stor erfaring med brug af computere. Vi regner dog med, at brugerne har prøvet at benytte et almindeligt tekstbehandlingsprogram tidligere. Input og output skal kunne hentes og gemmes på gængs måde via dialogbokse, eller kopieres direkte ind i programmet, fra f. eks. internettet, ved hjælp af konventionelle kopiér/indsæt metoder. Desuden skal det være mulig at skrive opskriften direkte ind det felt hvor teksten vises. Vi har valgt at input og output skal være i formen *.txt, og at tegnsættet skal være UTF-8. Dette er nødvendigt for at kunne håndtere tegn som f. eks $\frac{1}{2}$. Brugeren bliver altså nødt til enten, at konvertere filerne, der ønskes som input til dette format, eller at benytte kopiér/indsæt funktionen, der automatisk ændre kopieret tekst til *.txt.

Programmet skal have en fyldestgørende hjælpe-fil, således at brugerne kan bruge programmet og redigere i måleenhederne, uden at skulle have anden instruktion.

Opgaveformuleringen lægger stor vægt på, at programmet skal testes grundigt. Dette omfatter unit-test, glassbox-test og blackbox-test. Vi har desuden valgt at tilføje en mindre accepttest.

2.1 Planlægning.

På vores første gruppemøde startede vi med at se på opgaveformuleringen, og brugte den som udgangspunkt for vores kravspecifikation. Da dette var overstået, kunne vi gå i gang med at planlægge forløbet. Der blev lavet en tidsplan med seks milepæle som følger:

- M1 kravspecifikation
- M2 Design specifikation færdig
- M3 Kode og Unit test færdig
- M4 program/testfase færdig
- M5 dokumentation færdig
- M6 Deadline for aflevering

Ud fra disse kunne vi så få et overblik over arbejdsprocesserne, hvilke der var afhængige af andre, og hvilke der kunne udføres sideløbende. (se bilag B side 32)

2.2 Risikovurdering

Vi betragtede de ting vi mente kunne have indflydelse på vores projekt, og vurderede hver enkelt betydning som vist i nedenstående skema:

1. Sygdom - lav/medium
2. Hardware-fejl - meget lav
3. Software-fejl - lav
4. Fejl i vurdering af tidsplan - høj risiko
5. M1 kravspecifikation - meget lav
6. M2 Design specifikation færdig - lav
7. M3 Kode og Unit test færdig - middel
8. M4 program/testfase færdig - høj
9. M5 dokumentation færdig - høj
10. M6 Deadline for aflevering - meget lav

2.2.1 Uddybende bemærkninger

1. Sygdom. Vi mente ikke dette ville have den helt store indflydelse. Selv om en eller to af os skulle blive syge, så det ud til, at der ville være tid nok til at gennemføre de ting, vi gerne ville. I praksis viste dette sig at være korrekt, da et af gruppe medlemmerne blev syg i fem dage, uden at dette havde nogen indvirkning på vores tidsplan.
2. Hardware-fejl. Alle gruppemedlemmeren har mere en en computer til rådighed. Der ud over er der rigeligt adgang til computere på universitet. Ydermere benyttede vi os af CVS, så ved evt. nedbrud ville det højst blive de allernyeste versioner af koden der ville gå tabt, og derfor nem at rekonstruere.
3. Software-fejl. Disse blev også vurderet som værende en lav risiko. Vi kendte Eclipse i forvejen og betragter det som et godt og stabilt udviklingsmiljø.
4. Fejl i vurderingen af tidsplanen. Da ingen af os har megen praktisk erfaring at planlægge denne slags projekter, var der stor sandsynlighed, for at vi kunne vurdere tidsforbruget forkert. Det viste sig dog at planen holdt godt.
5. Milepæl 1. Da kravspecifikationen og tidsplanen blev lavet sammen, var der temmelig god mulighed for at overholde denne.
6. Milepæl 2. Her kunne vi let se at denne ville blive færdig til den fastsatte tid.
7. Milepæl 3. Vi havde et rimeligt overblik over den mængde arbejde dette ville medføre, og følte os derfor sikre på at vi kunne overholde tiden også her, dog med en lidt større usikkerhed end under de tidlige punkter.
8. Milepæl 4. Overholdelsen af denne og den næste milepæl, som jo ligger til sidst i projektet, er behæftet med den største usikkerhed. For det første kunne små overskridelser i de tidlige faser høbe sig op, og for det andet bevægede vi os her ind i de områder, hvor vi er mest uerfarne, specielt med hensyn til test.
9. Milepæl 5. Da risikoen for ikke at overholde den forrige milpæl var stor, følger det logiske at det er den også for denne.
10. Selve afleveringen var der ikke store risiko for at vi ikke overholder. Lige meget hvad vi havde nået, ville der blive afleveret til tiden.

Det viste sig at vores tidsplan holdt godt. Kun ganske små overskridelser fandt sted, og disse havde ingen indflydelse på det færdige resultat. For et skematisk overblik over milepælene og tidsplanen henvises til bilag B side 32.

Kapitel 3

Design

Vi startede forløbet med en gennemgang af hvordan et kageopskrifts konverteringsprogram ville kunne bruges, og udarbejdede i denne forbindelse to aktivitetsdiagrammer over mulige brugsscenerier (se bilag A side 30). Vi besluttede hurtigt, at programmet skulle have en grafisk brugerflade, således at målgruppen af kagebagende husmødre vil kunne bruge programmet, uden at skulle bruge kommandolinien. Enkelte hurtige skitser om programmets visuelle fremtoning blev lavet, men grundet det meget simple udseende, forblev de løse skitser og er ikke medtaget i denne rapport. Det skal dog nævnes at programmet ser ud som dengang aftalt.

Vi lagde os fast på en iterativ udviklingsmodel, således at vi kunne justere krav- og design-specifikationer efterhånden som problemstillinger opstod, noget vi har benyttet os af gennem forløbet.

Designet aftaltes til at skulle bestå af et hoved vindue, der skulle varetage konverteringen, og et undervindue, der skulle stå for redigeringen af nye omregningsenheder. I hovedvinduet skulle der oprettes knapper til at hente filer, gemme filer, konvertere og en knap til at åbne undervinduet. I sidste øjeblik indførtes, på baggrund af accepttesten, en knap til at slette alt teksten i tekstfeltet. I undervinduet, skulle der være felter til at indtaste nye enheder, samt knapper til at tilføje enheder, slette enheder, nulstille felterne, og en luk knap.

Det skal i denne forbindelse nævnes, at programmet ikke tager højde for tal skrevet som bogstaver (eks. twenty inches), ligesom stavfejl heller ikke håndteres (eks. 15 Ib, hvor der er skrevet et stort "i" i stedet for lille "l"). Det blev også besluttet, at omregningsfaktoren skulle være et komma-tal, hvorimod konstant faktoren kunne nøjes med at være et heltal. Sluteligt blev vi enige om ikke at tage højde for 'volapyk', f.eks. hvis brugeren forsøger at omregne -500 Fahrenheit, en temperatur der ligger under det absolutte nulpunkt, vil programmets svar ligeledes være urealistisk (-295,79 Celsius).

For at udnytte tiden optimalt, delte gruppen arbejdet op i to dele, en programme rings del og en test del. Denne opdeling gjorde, at vi samtidigt kunne programmere test og programkode, idet vi havde besluttet funktionernes interface.

3.1 Tekniske design beslutninger

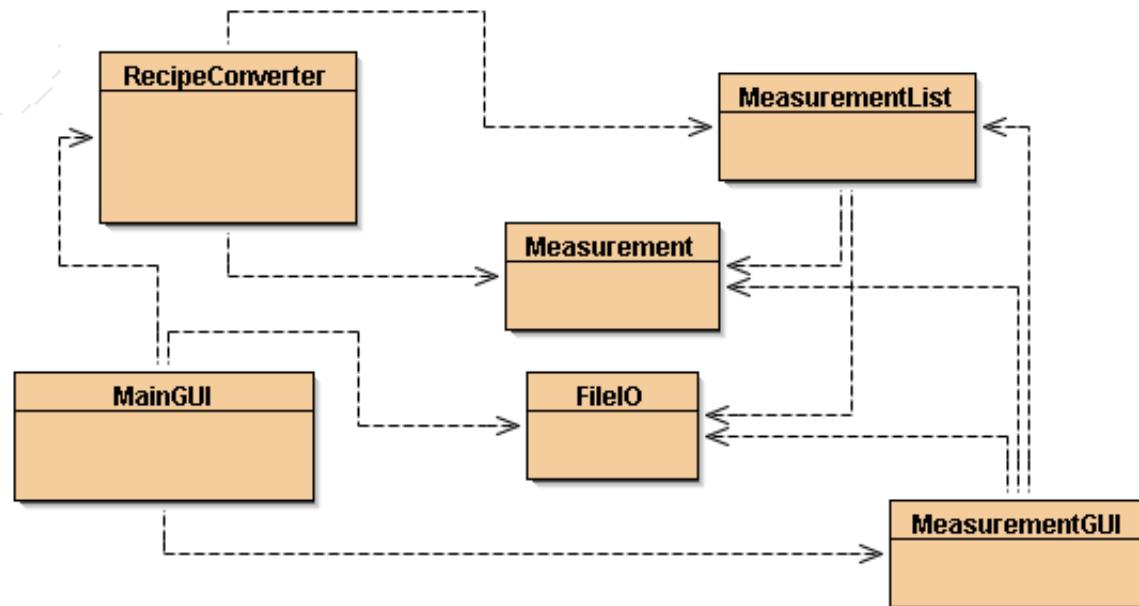
Da opgaven skulle programmeres i Java, besluttede vi at bruge Eclipse som editor, Java version 5, og Eclipses indbyggede CVS styring, for hurtigt at kunne koordinere og distribuere egen arbejdsindsats.

I forbindelse med testningen fandt vi ud af, at tal-formatet 'float' ikke var præcist nok, men skabte en afskæringsfejl (eng.: truncation error) på grund af totalssystemets svaghed med uheldige brøkdele, der ikke er delelige med $\frac{1}{2^n}$. Hvis vi brugte 'float' ville $2,54 * 2,75 = 6,9849997$, hvor det korrekte er 6,985. Derfor benyttes formatet 'double' der indeholder nok bits til at kunne beregne resultatet præcist.

I almindelig dansk matematik afrundes et 5-tal opad. I Java benyttes en afrunding der hedder 'round even'. Det betyder at et 5-tal afrundes mod det lige af de to heltal det er placeret imellem. 10,095 afrundes dermed til 10,09 i stedet for 11,00. Vi har valgt ikke at tage yderligere notits af dette, da vi vurderede at det rent gastronomisk ikke vil have nogen betydning.

Kapitel 4

Implementering



Figur 4.1: Programmets klasser og struktur.

4.1 Measurement

Denne klasse repræsenterer én omregningsform, fra én enhed til én anden enhed. Klassen indeholder fire attributter `from`, `to`, `factor` og `shiftFactor`.

En instans kan initialiseres ved at give konstruktøren værdier til disse fire attributer. Kan benyttes til indlæsning fra et serialiseret lager.

Klassen tilbyder metoden `convert()`, som givet en streng som input - konvertérer inputtet til tal, og derefter omregner, ved at lægge `shiftFactor` til og gange med

factor. Alle omregninger foregår således på formlen:

$$amount = (amount + shiftFactor) \cdot factor.$$

Herefter formatters resultatet til et dansk tal-format med komma som decimalseperator og punktum som tusind-seperator. Tallet afrundes desuden til et maksimum af to decimaler, som ifølge design-afsnittet.

I Measurement findes desuden en statisk metode kaldet `convertToNumber()` som, givet en streng, konverterer denne til en `double`. Denne funktion håndterer ASCII-specialtegn så som $\frac{1}{2}$, $\frac{1}{4}$ og $\frac{3}{4}$ men også Unicode-tegnene for $\frac{1}{3}$, $\frac{2}{3}$, $\frac{1}{8}$, $\frac{3}{8}$, $\frac{5}{8}$ og $\frac{7}{8}$. Dette gøres ved at undersøge om inputtet for hvert af disse tegn eksisterer i strengen, og derefter lægge brøkens værdi til det endelig resultat. Tilslut fjernes specialtegn fra strengen, og den resterende streng konverteres til `double` v.h.a. Java's indbyggede funktion til dette formål.

Derudover indeholder `Measurement` en funktion `toString()`, og en tilhørende konstruktør, der tager en streng. Disse tilbydes hhv. at serialisere og deserialisere attributter og benyttes af `MeasurementList` til at gemme alle omregningsformer i en fil, og tilsvarende genoprette `Measurement`-objekterne udfra en fil.

4.2 MeasurementList

`MeasurementList` holder styr på alle `Measurement`-objekter og er den klasse, som tilbyder funktioner til at tilføje nye, fjerne eller finde en omregningsmetode (et `Measurement`-objekt) udfra en given enhed. Dette gøres vha. funktionen `getMeasurement()`. `MeasurementList`, der automatisk gemmer og henter alle metoder til og fra én fil på disken, således at ændringer gøres persistente mellem forskellige programkørsler. Der hentes i det øjeblik `MeasurementList`-objekt initialiseres, og der gemmes ved hvert kald til `addMeasurement()` og `removeMeasurement()`.

Lagring af listen foregår ved at kalde funktionen `toString()`, som er implementeret ved at kalde `toString()` på hver `Measurement`-objekt i listen og adskille disse med linjeskift. Når der gemmes udskrives listen med systemets linjeskift, og er derved afhængig af om programmet køres på UNIX, Windows eller Mac. Dette håndteres dog ved indlæsning i `load()`, hvor filen deles på ét eller flere af enten tegnene CR (cursor return) eller LF (ny linje).

Der tilbydes desuden en funktion `getMeasurementTypes()` til at returnere alle, på et givet tidspunkt, enheder, der kan konverteres fra af listen.

Et map over alle metoder er internt i `MeasurementList` på alle tidspunkt holdt med fra-enhed, med småbogstaver, som nøgle.

4.3 RecipeConverter

Denne klasser tilbyder én funktion, `convert()`, som tilbyder konvertering af enheder i en hel opskrifttekst. Funktionen, i sin endelig udgave, implementerer ved internt at kalde tre private funktioner. Disse foretager hver i sær en søg-og-erstat i teksten udfra søgekriterie der defineres i et regulært udtryk (eng: regular expression). Funktionerne benytter alle en privat funktion kaldet `getMeasurementPattern()` til at konstruere deres regulære udtryk. Denne funktion samler og definere en række oplysninger, som bruges til at basere søgeresultaterne på. Følgende defineres:

- `hitBefore` definerer, at et givent søgeresultat skal, enten være først i tekst-strengen, eller lige efter et mellemrum, linjeskift, eller tabulator-tegn.
- `hitNumbers` definerer, at vi ved et nummer forstår et eller flere tal efterfulgt af hinanden, evt. med et mellemrum, og evt. indeholdende punktum, eller et eller flere brøktegn, som `Measurement.convertToNumber()` kan håndtere.
- `hitSpaces` definerer, at vi ved et mellemrum (mellem tal eller enhed) forstår et eller flere mellemrum, linjeskift, tabulator, et gradetegn og/eller en bindestreg.
- `hitAfter` definerer, at et givent søgeresultat skal optræde lige før slutningen af tekst-strengen eller lige før et mellemrum, linjeskift, tabulator-tegn, punktum, komma, semikolon, kolon eller bindestreg.
- `hitUnits` definerer, at der skal optræde én af de konverteringsenhed, som vi kan konvertere fra, defineret af `MeasurementList.getMeasurementTypes()`.

Første funktion, der kaldes i `convert()`, er `convertNumberXNumber()` - denne funktion søger efter tekst, hvori der optræder to numre adskilt af enten "x" eller "*" og efterfulgt af en af de fra-enheder, som vi kender. Funktionen konverterer herefter, for hvert søgeresultat, de givne tal og erstatter enheden med tilsvarende til-enhed v.h.a. `Measurement`.

Herefter kaldes `convertUnitNumber()` som foretager en søgning, hvori der først optræder en kendt fra-enhed, derefter et tal, men dog ikke efterfulgt af endnu en af de enheder, som vi kender. Denne hindring er tilføjet for at undgå at oversætte enheder, så som i "put in 20 oz of sugar". Funktionen konverterer herefter, for hvert søgeresultat, måleenheden til den tilsvarende til-enhed og bytter nu om på enhed og tal således at f.eks. "Gas mark 5" omskrives til "190 °Celsius".

Til slut kaldes `convertNumberUnit()`, som foretager en søgningen på tal efterfulgt af en kendt fra-enhed. Disse oversættes derefter til den tilsvarende til-enhed.

Fælles for alle funktionerne er at mellemrum/linjeskift, m.v., der tidligere optrådte mellem, før og efter måletal og måleenheder bibeholdes.

4.4 Ændringer igennem udviklingsforløbet

Igennem de tre iterationer, der er blevet gennemført, har tests medført, at en række ændringer i programstrukturen, blev nødvendige. I dette afsnit beskrives kort udseende af programmet gennem forløbet.

I første iteration var der lagt op til, at programmet kun indlæste tekst fra et ASCII-formatterede filformat. Det viste sig i midlertidigt at special tegn allerede fandtes i de allerførste test-filer. Derfor måtte dette revideres og ændres til UTF-8.

Efter første iteration indeholdt programmet kun support for konvertering af en enhed, hvori måletalet optrådte netop før en kendt enhed og derfor blev måletal, som f.eks. "gas mark 5" og "12.5 x 8.5 inch" hhv. ikke omregnet, og ikke omregnet korrekt.

I anden iteration blev der tilføjet support for den modsatte notation for alle enheder, samt tilføjet support for sammenhørende måletal opdelt med enten gange-tegn eller x. Ligeledes blev en fejl, der gjorde at systemet ikke håndterede enheder skrevet med storrebogstaver korrekt, rettet. Dette blev gjort ved, at ændre mappet i MeasurementList til altid at være med fra-enhed med småbogstaver.

Efter anden iteration blev der fundet enkelte placerings problemer samt afrundingsfejl. Derudover blev det fundet, at det var uhensigtsmæssigt, at programmet altid gemte opskrifter med UNIX-nylinjeskift. Ydermere blev det fundet at typen float i Java forårsagede en mindre afrundings-uhensigtmæssighed.

I tredje iteration blev lagringstypen skiftet til double og der blev afprøvet yderligere på de benytte regulære udtryk (eng.: 'Regular expressions') i RecipeConverter, som følge af tests. Derudover blev der tilføjet afrunding af omregnede måletal i Measurement.

I sidste øjeblik blev det opdaget at brøkdele skrevet som division f.eks. 1/4 eller 3/4 ikke blev konverteret. Heller ikke mål skrevet i parantes blev fanget. Dette er også blevet rettet.

4.5 Indlæsning og skrivning af filer

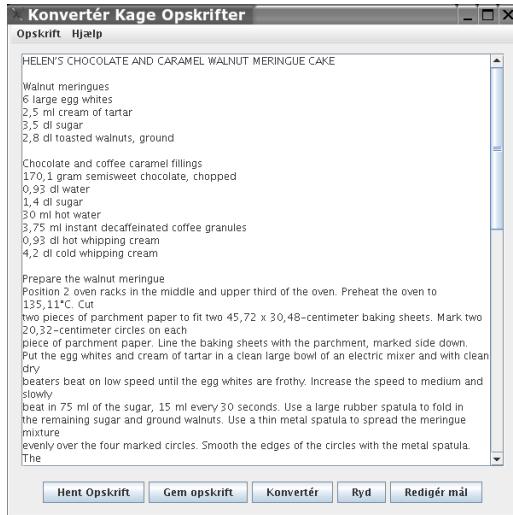
Klassen FileIO varetager indlæsning og skrivning af filer til og fra programmet. Denne har to statiske metoder, public static String readFile(File text) og public static void writeFile(File file, String text).

Den første, readFile() benytter Scanner-klassen til at indlæse en *.txt fil én linie ad gangen, fører dem sammen med et linjeskift, og returnere den som en streng. Desuden sættes tegnsættet til UTF-8. Hvis den valgte fil ikke er i *.txt-format, eller det er en tom fil, gives der en fejlmeldelse, der fortæller dette.

Den anden metode writeFile() benytter OutputStreamWriter til at skrive den valgte tekst til en fil i *.txt-format.

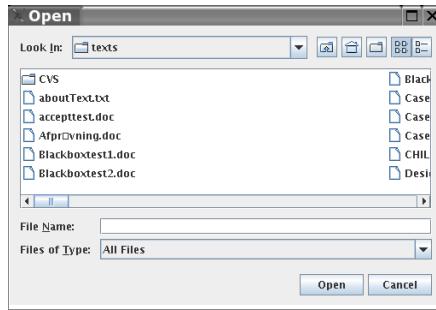
4.6 Brugergrænseflade

Brugergrænsefladen består af to vinduer. Hovedvinduet, der vises når programmet startes, og redigerings vinduet, der åbnes når brugeren ønsker at tilføje eller ændre måleenheder. Hovedvinduet er bygget op med en menu-bjælke øverst, et stort felt til at vise de opskrifter, der indlæses eller skrives i midten, og fem knapper, som tilbyder programmets funktionalitet, nederst. Vinduet er sat til en fast minimumstørrelse, når det åbnes, så det på den måde sikres, at alle knapperne vises. Desuden er det centreret på skærmen.



Figur 4.2: Hoved vinduet med en konverteret opskrift

Menu-bjælken har to menuer, opskrift-menuen og hjælpe-menuen. Opszrift-menuen tilbyder stort set den samme funktionalitet som knapperne i bunden af vinduet, men adskiller sig dog ved ikke at have muligheden for at rydde vinduet, og desuden er her muligheden for at lukke programmet. Hjælpe-menuen tilbyder hjælpefilen, og "om-filen, der kort fortæller om programmet og programørerne. Funktionaliteterne "Hent opskrift" og "Gem opskrift", der tilbydes både via menuen og via knapperne, benytter sig begge af metoder, der kalder `FileIO`-klassen. Der åbnes gængse dialogbokse, der lader bruger vælger filer og biblioteker. Som nævnt under beskrivelsen af `FileIO`, vises fejlmeddelser hvis de valgte filer ikke svarer til det krævede format.



Figur 4.3: Åben-dialogbox

Menupunktet "Konvertér opskrift" og knappen "Konvertér" kalder programmets hovedfunktionalitet. Dette gøres ved, ved hjælp af klassens egen convert() -metode, der sender teksten fra tekstfeltet, som parameter til metoden convert() i klassen RecipeConverter. Denne returnerer teksten, nu med de metriske enheder. Teksten vises så i tekstfeltet. Hvis man ønsker at ændre eller tilføje måleenheder, kan man åbne redigeringsvinduet enten fra "Opskrift-menuen, eller ved hjælp af "Redigér mål-knappen.

Fra:	Faktor:	Til:	Konstant:
yard	m	0.914399	0
tablespoons	ml	15.0	0
cups	dl	2.8	0
o	l	0.568	0
mile	km	1.609342	0
inches	centimeter	2.54	0
quart	liter	1.1	0
ounce	gram	28.35	0
ounces	gram	28.35	0
cm	cm	1.0	0
teaspoon	ml	5.0	0
pole	m	5.0292	0
pound	gram	454.0	0
Gallon	l	4.546	0
Gas mark	*Celsius	12.75	10
Fahrenheit	Celsius	0.556	-32
pint	l	0.568	0
oz	g	28.35	0
cup	dl	2.8	0
feet	m	0.3048	0
Inch	centimeter	2.54	0
C	C	0.556	-32
tablespoon	ml	15.0	0
Gasmark	Celsius	12.75	10
lb	g	454.0	0
teaspoons	ml	5.0	0

Figur 4.4: redigeringsvinduet med tabellen over måleenheder

Dette vindue har øverst fire små felter, hvor man kan skrive navnet på den enhed, som skal tilføjes, dens omregnings faktor, navnet på den enhed den omregnes til, og en evt. konstant til brug i omregningen. Under disse felter er der et stort felt, der viser en tabel med de mål programmet kan konvertere. Det er ikke muligt at redigere i dette vindue, dog kan der kopieres fra det. Fra knapperne forneden kaldes de metoder, der fører mål til eller ændrer dem. Når dette sker, laves der først et type-check for at sikre

at felterne er udfyldt korrekt, der efter udføres selve ændringen, og til sidst opdateres tekstuelt med tabellen.



Figur 4.5: fejlmeldelse ved forkert indtastning i redigeringsfelterne

Ønsker man at slette, åbnes der en dialogboks, hvor man skriver navnet på den enhed, der skal slettes. Når man derefter klikker på "ok" udføres sletningen og tekstuelt opdateres. Udenfor "tilføj" og "slet" knapperne, er der også en knap til at rydde felterne, så det er let at starte på ny. Denne funktion kaldes i øvrigt automatisk når en ændring udføres.



Figur 4.6: Inputboks til sletning

Generelt om brugergrænsefladen kan det siges, at vi ønskede at lave et design, der var så enkelt og konventionelt som muligt, således at uvante computerbrugere ikke kommer på glatis, når de anvender programmet.

Kapitel 5

Afprøvning

Da det blev besluttet at bruge en iterativ udviklingsproces, så har afprøvning været et stadigt tilbagevendende tema i projektet.

For at få et større fokus på afprøvningen af programmet, besluttet vi at dele opgaverne i projektet, så den ene halvdel af gruppen skulle stå for udviklingen af selve programmet, og den anden halvdel for udviklingen af testene. På den måde håbede vi at gøre testene og programmet bedre. Den halvdel af gruppen, som tog sig af afprøvningen, blev på den måde en ekstern testgruppe.

Efter at vi valgte at dele opgaverne, gik der prestige, for testerne, i at finde fejl, så de, som udviklede programmet, fik mange tilbagemeldinger, om at der til stadighed blev fundet fejl, som der skulle rettes.

Da vi var blevet enige om kravspecifikationen, og havde besluttet designet af løsningen, tog vi fat på at udvikle både programmet og testene. Da første udkast af programmet var færdig, kunne vi umiddelbart starte med afprøvningen. Vores forhåbning var, at ved at teste tidligt, ville vi nemmere kunne slippe af med de fleste småfejl, og også nemmere kunne benytte den iterative proces, og gå tilbage og se om vores design var optimalt. Tidligt i forløbet er det også nemmere at foretage ændringer i koden.

Da vi startede med at udvikle testene, samtidig med at vi udviklede koden, valgte vi at starte med unit-test og blackbox test. Vi behøver ikke at kende koden, for at lave sådanne test.

Da første udkast af programmet var færdig, tog vi fat på afprøvningen. Vores blackboxtest skulle stadig udvikles mere, for ved brug af den iterative proces kan ikke alt testes fra starten af.

Den første blackboxtest skulle tage sig af åbenbare fejl på konvertering af teksten, mens unit testene skulle tage sig mere detaljeret af funktionerne, og samtidig også få disse testet for logiske fejl.

Når de første fejl var afsløret ved hjælp af disse test, skulle vi så rettet programmet til, og korrigere vores design. Denne proces ville så fortsætte i følge vores tidsplan. (Se tidsskema bilag B side 32).

Senere, når vi var færdige med at rette op på fejlene fra blackboxtesten og unit-testen, eller at vi havde nået tidsgrænsen for denne del, ville vi så udvikle en whiteboxtest til

gennemtestning af centrale dele af programmet.

Dette skulle gøres ved at studere koden, for gennem en undersøgelse af programmets opbygning, at finde den vigtigste del af programmet, selve kernen, og gennemteste denne for alle mulige fejl. Selve kernen i programmet er jo netop den del, der finder det i teksten, der skal konverteres, og som udfører selve konverteringen.

Whiteboxtesten skulle derfor teste netop denne del af koden.

5.1 Test af ekstra opskrifter

Til sidst i forløbet har vi testet vores program med en lang række opskrifter udover dem, som er givet i opgaven. Blandt andet fire, der blev fundet på hjemmesiden nævnt i opgaven.

Opskrifterne hedder Dirt Cake 1, Lord Baltimore Cake, Maraschino Cherry Nut Cake og Large white Birthday Cake. (De findes i bilag D side 37)

Ved første afprøvning så fejlede konverteringen af flere enheder på alle opskrifterne. Dette skyldtes, at der ikke havde tages højde for tal skrevet som f.eks. $1/2$, $2/3$, $1 \frac{3}{4}$, og heller ikke tal med mål skrevet inden i parenteser. F.eks. (5 ounces).

Vi besluttet derfor at disse skulle konverteres, da der hyppigt optræder sådanne tal, og parenteser i opskrifterne.

Vi skrev derfor om i koden, og vi kunne se at disse opskrifterne blev konverteret uden problemer, da programmet blev testet igen.

5.2 Grænsetilfælde

Når det gælder tal, der står foran eller bag enheden i opskriften, så kan det være et hvilket som helst tal. Da vi ikke tager højde for fejl i opskriften, konverterer programmet tallet ukritisk (f.eks. Gasmark -4, dette giver ingen mening på engelsk, så det vil heller ingen mening give på dansk).

Der er ikke bestemt ækvivalensklasser og testet for grænser på disse tal, da programmet ikke kan rette fejl, der måtte findes i opskriften, og man må derfor påregne, at der i disse tilfælde kommer forkerte tal ud.

Programmet ved intet om ækvivalensklasser for de forskellige måle-enheder, derfor må brugeren altså komme med korrekt input før man kan forvente et korrekt resultat.

Vi benytter en begrænsning på størrelse af tal til typen "double" i vores program. Dette giver en max-værdi på $(1.7977) \cdot e^{308}$ og en min-værdi på $(-1.7977) \cdot e^{308}$. Vi antager, at der er en fejl i opskriften, hvis tallet er højere eller lavere end disse grænser.

5.3 blackboxtest

Udviklingen af blackboxtesten startede lige efter, at kravspecifikationen og designspezifikationen var blevet fatlagt.

Blackboxtesten fulgte udviklingsprocessen og blev dermed omskrevet 3 gange. Den anden og den tredje blackboxtest indeholdt alligevel alle test fra de foregående iterationer, for at sikre at test, der før havde bestået, ikke ville dumpe.

Den første blackboxtest blev udviklet til at afprøve den del af programmet, der indlæser en tekst og konverterer målene fra britisk til metrisk. Blackboxtestene indeholder testnummer, testtilfælde, forventet resultat, resultat og evaluering.

Under testtilfælde er der beskrevet, hvad der skal testes for. Testtilfældene blev udviklet med tanke på, at finde fejl som programmet ikke kunne håndtere, fejl som udvikleren af programmet ikke havde fundet ud af, åbenbare fejl, og uhensigtsmæssigheder som udvikleren ikke karakteriserer som fejl, men som opleves som fejl for brugeren.

Forventet resultat er det, testerne forventede ud fra designspezifikationen, og hvad det forventede logiske eller matematiske resultat var.

Det faktiske resultatet af testen blev så nedskrevet, og til slut i testen blev der givet en evaluering, om testen bestod eller dumpede. Evalueringen af testen blev så lagt frem for hele gruppen til diskussion.

Da det ikke var besluttet i vores designspezifikation, hvordan vi skulle håndtere afrunding af konverterede tal, blev de forventede resultaterne i den første blackboxtest bestemt ud fra logisk tankegang, om hvad en bruger af programmet synes var læseligt. Dette viste sig at dumpe rigtig mange af testtilfældene. Vi måtte derfor gå tilbage til designspezifikationen og træffe nye beslutninger.

Alle fejl blev udbedret, og udviklingen af anden del af programmet kunne starte. Anden del af programmet er den del hvor brugeren kan tilføje nye mål, som programmet skal kunne benyttes sig af.

Blackboxtest nummer to blev også opdateret iht. designspezifikationen. Der blev også tilføjet nogle flere testtilfælde, da programmet var blevet yderligere udviklet.

Test nummer to blev så udført. Mange af de forventede resultater blev ændret før denne test.

Evalueringen af blackboxtest nummer to blev lagt frem for gruppen og diskuteret. Resultaterne af denne test var rigtig gode, men der var fortsat nogle fejl, som ikke var rettet. Resultaterne diskuteredes grundigt i gruppen for at beslutte den videre udbedring. Det blev besluttet at også de sidste fejl skulle udbedres, da der fortsat var tid tilbage for denne del tidsplanen.

Blackboxtest nummer tre blev så opdateret iht. beslutninger. Testen blev kørt igen nem og resultatet var at samtlige test bestod. Da vi også har unit-test og whiteboxtest på de vigtigste funktioner i vores program, betragtede vi derfor blackboxtestene som vellykkede og udførte ikke flere test med den metode.

Slutresultat af blackboxtestene er, at alle test er bestået, og vi mener at have fået afdækket rigtig mange fejl, og også rettet op i de fejl vi har fundet.

5.4 unit-test

Så snart design specifikationen lå klar, herunder programmetodernes interface, startede programmeringen af unit-test. Efterhånden som design beslutninger blev truffet, som f.eks. afrunding, hvor vi fandt ud af, at vi ikke havde den fornødne præcision (se mere under design), blev testene tilpasset de nye forventede resultater.

Unit testning af programmet blev udført ved hjælp af J-Unit. Vi valgte ikke at køre unit-test på de metoder, der har med den grafiske brugerflade at gøre. Denne del blev intensivt testet med Blackbox testning, og er desuden meget visuelt betonet, noget det menneskelige øje er bedre til at afgøre. De klasser, der er blevet testet med J-Unit, er `FileIO.java`, `Measurement.java`, og `RecipeConverter.java`.

Det er i særdeleshed metoden `convert` i klassen `RecipeConverter`, der er blevet unit-testet, for at afklare om den kunne håndtere følgende:

- Punktummer
- Store og små bogstaver
- Whitespace
- Paranteser
- Komma
- Rækkefølge af enhed og tal
- Brøktal
- Dobbelt omregninger med * imellem
- Fra og til beregninger med bindestreg
- Helt korte strenge
- Strenge der ikke skal konverteres noget i

Også `convertToNumber` i klassen `Measurement`-klassen, er blevet unit testet, for at undersøge om den kan håndtere UTF-8 specielle brøktegn, brøker med "/" og sammensatte brøker med begge dele.

J-Unit anvendte vi også i forbindelse med white-box testen, idet det er hurtigt at teste med, når testen én gang er lavet; computeren laver alt arbejdet, i modsætning til en manuel indtastet blackboxtest.

Efter udviklingen af programmet består det alle de vedlagte unit-test, og det må derfor konkluderes, at denne del har været succesfuld, med hensyn til at få rettet fejl i programmet.

5.5 whiteboxtest

Da vi med blackboxtestene og unit testene fik testet hele programmet uden at rigtigt kende koden, har vi valgt at teste kernen af programmet med en whiteboxtest.

Kernen i programmet er den klasse som hedder `RecipeConverter`, og den består af 3 kernefunktioner, så vi besluttede at lave en test for hver af funktionerne i klassen, hvor vi skal teste for alle mulige fejl. Kernefunktionaliteten i programmet består i at finde alle de tilfælde i opskriften, hvor der står britiske mål, og så konvertere disse til metriske mål. Klassen `RecipeConverter` har en funktion som opbygger et regulært udtryk (se program beskrivelsen for denne) ud fra navne på de kendte britiske enheder.

Vi besluttede at lave 3 Case tilfælde. De 3 case tilfælde vi søger på er, hvor der står et tal efterfulgt af en enhed, hvor der står en enhed efterfulgt med et tal, og hvor der står et tal gange et tal efterfulgt med en enhed.

Det mest almindelige tilfælde er, hvor vi søger efter et tal med en enhed bagefter, de to andre tilfælde kan betragtes som specialtilfælde.

5.5.1 Case 1

Dette er et af de to specialtilfælde. Det vi undersøger her er tilfældet hvor der står to tal, adskilt af gange eller `x`. Det der ledes efter er strenge bestående af 7 elementer, hvor vi er specielt interesserede i de 5 midterste elementer som regulær ekspression. fx når der søges efter: `2 x 2 inch`

De 7 elementer med regulære udtryk afgrænsner vores ækvivalensklasser, og de bliver da følgende for case 1:

1. Det der skal stå før strengen, der skal findes:
 - Gyldige: mellemrum, tab andet whitespace eller liniestart
 - Ugyldige: Alt andet
2. Nummer, evt. med punktum
 - Gyldige: nummer
 - Ugyldige: tekst eller tegn
3. Gangetegnet
 - Gyldige: mellemrum \times mellemrum eller mellemrum $*$ mellemrum
 - Ugyldige: alt andet
4. Nummer, evt. med punktum
 - Gyldige: nummer
 - Ugyldige: tekst eller tegn

5. Mellemrum.

- Gyldige: mellemrum, tab, andet whitespace linieskift
- Ugyldige: alt andet

6. Enhed

- Gyldige: én af de enheder, der står i listen
- Ugyldige: alt der ikke står i konverteringslisten

7. Det, der skal stå efter strengen

- Gyldige: whitespace, punktum, komma, kolon, semikolon, bindestreg eller slutning
- Ugyldige: Alt andet, fx bogstaver

Der er 128 forskellige kombinationer, vi udelader derfor nogen af dem, der minder meget om hinanden, og fejler på flere poster

Visse af testene, vil ikke konverteres af case 1, men vil delvist konverteres af case 2, dette tager vi højde for.

5.5.2 Case 2

Dette er det andet af de to specialtilfælde. Det vi undersøger her er tilfælde, hvor der står et tal efter en enhed. Det, der ledes efter er strenge bestående af 6 elementer. Et eksempel er når vi søger efter: Gas mark 2

De 6 elementer med regulære udtryk afgrænsner vores ækvivalensklasser, og de bliver da følgende for case 2:

1. Det der skal stå før strengen, der skal findes:

- Gyldige: mellemrum, tab andet whitespace eller liniestart
- Ugyldige: Alt andet

2. Enhed

- Gyldige: én af de enheder der står i listen
- Ugyldige: alt der ikke står i konverteringslisten

3. Mellemrum.

- Gyldige: mellemrum, tab, andet whitespace linieskift
- Ugyldige: alt andet

4. Nummer, evt. med punktum

- Gyldige: nummer
 - Ugyldige: tekst eller tegn
5. Det der skal stå efter strengen
- Gyldige: whitespace, punktum, komma, kolon, semikolon, bindestreg eller slutning
 - Ugyldige: Alt andet, fx bogstaver
6. Det der skal stå efter post 5
- Gyldige: whitespace, punktum, komma, kolon, semikolon, bindestreg, slutning eller en enhed vi ikke kender
 - Ugyldige: et mellemrom efterfulgt af en enhed vi kender.

For case 2 er der 64 forskellige kombinationer. Vi har lavet en Junit test som dækker alle de forskellige kombinationer, og de er alle bestået, da vi har besluttet at medtage en forudsætning. Denne er, at hvis post 6 fejler, så vil enheden, som står opgivet bag tallet, blive konverteret, og ikke enheden som står foran. Dette sker kun, hvis den enhed som kommer til slut har sit tal efter enheden, og det er et specialtilfælde. Gruppens beslutning var, ikke at lave om på dette, da det virker usandsynlig, at dette skulle ske. Alligevel blev denne fejlen opdaget af whiteboxtesten, og vi vil gøre rede for dette i brugervejledningen.

5.5.3 Case 3

Dette er det mest almindelige af de tre tilfælde. Det vi undersøger her er tilfælde, hvor der står et tal foran en enhed. Det der søges efter er strenge bestående af 5 elementer. Et eksempel er når vi søger efter: 2 ounce

De 5 elementer med regulære udtryk afgrænsrer vores ækvivalensklasser, og de bliver da følgende for case 3:

1. Det der skal stå før strengen, der skal findes:
 - Gyldige: mellemrum, tab andet whitespace eller liniestart
 - Ugyldige: Alt andet
2. Nummer, evt med punktum
 - Gyldige: nummer
 - Ugyldige: tekst eller tegn
3. Mellemrum.

- Gyldige: mellemrum, tab, andet whitespace linieskift
- Ugyldige: alt andet

4. Enhed

- Gyldige: én af de enheder der står i listen
- Ugyldige: alt der ikke står i konverteringslisten

5. Det der skal stå efter strengen

- Gyldige: whitespace, punktum, komma, kolon, semikolon, bindestreg eller slutning
- Ugyldige: Alt andet, fx bogstaver

For case 3 er der 32 forskellige kombinationer. Vi har lavet en Junit test som dækker alle de forskellige kombinationerne, og de er alle bestået.

Der er også taget højde for i vores program, at der skal kunne hentes rigtig store opskrifter, eller mange opskrifter, som kan konverteres samtidig. Grænsen for hvor stor en opskrift kan være afhænger af hukommelsen på den maskine som programmet kører på.

5.6 Accepttest

Accepttesten blev udført af en enkelt bruger. Brugeren er, en ældre dame midt i 70'erne, som holder af at bage kager. Brugeren har en søster, der bor i England. Denne sender af og til opskrifter pr. e-mail, så programmet har en god relevans. Hendes erfaring med computeren er på et meget let brugerniveau, og strækker sig til at kunne bruge et tekstbehandlingsprogram, at hente og læse e-mails, og at benytte internettet.

Brugeren havde umiddelbart ingen problemer med at bruge programmet. Indlæsning af opskrifter blev udført på tre forskellige måder: Først ved at hente en fil direkte ved hjælp af fil-dialogen, så ved at kopier en tekst fra et andet medie (tekstbehandlingsprogram), og ved at skrive en opskrift direkte ind i tekstfeltet. På samme måde var der heller ikke nogen problemer med at gemme de konverterede opskrifter. Også tilføjelsen af nye måleheder, og redigeringen af gamle, foregik problemfrit.

Angående funktionaliteten af programmet havde brugeren en enkelt ting at bemærke: Der manglede en knap (funktionalitet) til at rydde det store tekst felt på en nem måde.

Selve konverteringen blev kritiseret for at ændre 'teaspoon' (og 'tablespoon') til liter-mål, og ikke til 'teske' (og 'spiseske'). Dette var på ingen måde hjælpsomt. På det tids-punkt i udviklingen hvor programmet blev testet, blev 'teaspoon' (og 'tablespoon') om-regnet til centiliter. Hvis dette skulle give nogen mening, skulle det i det mindste være til milliliter.

Generelt var brugeren tilfreds med programmets brugbarhed, men påpegede, at det måske var mere relevant, at konvertere fra Amerikanske mål. Britiske opskrifter af nyere dato, er næsten altid opgivet både i britiske og metriske mål. Det kan tilføjes at brugeren ser frem til at modtage en kopi af det færdig program, og takkede på forhånd gruppen med en kage.

Resultatet af testen blev givet mundtligt, så vi har desværre ikke noget skriftligt at vedlægge som bilag.

På baggrund af denne test, blev det besluttet at tilføje en 'Ryd'-knap til brugergrænsefladen, og at rettet 'ske'-målet.

Kapitel 6

Brugervejledning

Følgende vejledning er identisk med programmets hjælpefunktion.

Programmet kan konvertere opskrifter fra britiske mål til danske.

Opskriften kan indtastes i tekstfeltet, de kan kopieres og indsættes eller de kan indlæses fra en fil, ved tryk på 'Hent opskrift'. Når man bruger 'Hent opskrift' funktionen skal teksten have formatet *.txt.

Når opskriften står i det hvide tekstfelt, konverteres opskriften ved at trykke på 'Konverter'.

Opskriften kan herefter gemmes, ved tryk på 'Gem opskrift'.

Hvis der er måleenheder, der ikke umiddelbart konverteres af systemet, undersøg da om de står i konverteringslisten, der kan åbnes ved tryk på 'Rediger mål'. Hvis der er stavfejl eller engelske mål, der ikke giver 'giver mening' (f.eks. Gasmark -4, der ikke eksisterer, eftersom skalaen går fra 1 til 8) vil programmets oversættelse heller ikke være korrekt.

Der er en enkelt anden mulighed for fejl, hvis der står en enhed foran et tal, og lige herefter en enhed foran et tal, konverteres de to midterste, hvilket ikke er korrekt. Vi håber ikke dette vil have nogen betydning, idet det er højest usandsynligt at det vil ske i en opskrift.

Alle enheder kan redigeres. Bemærk at programmet fra start konverte-rer fra britiske mål, ikke amerikanske. Ønskes dette ændret, eller hvis man bare gerne vil have målene opgivet på en anden måde, kan man selv ændre målenes faktor ved at overskrive de engelske som anført nedenfor.

For at tilføje et mål til listen, klik da på 'Redigér mål', og i det nye vindue indtast det nye mål i felterne: 'fra', 'faktor', 'til', og 'konstant' øverst. Hvis et mål eksistere overskrives det, ellers tilføjes det til listen. 'Fra' og 'til' skal være bogstavfelter, faktor skal være et tal, hvis det er et decimaltal, brug venligst punktum i stedet for komma. Konstant skal være et helt tal.

Hvis man ønsker at slette et mål, kan 'Slet' funktionen anvendes.

God fornøjelse med bagningen!!

Kapitel 7

Konklusion

Vi har løst hele opgaven. Vi har lavet en præcis kravspecifikation på grundlag af en analyse af opgavenformuleringen. Dette er brugt til et fyldestgørende design. Dette har igen gjort det nemt at implementere og teste programmet. Der er udført grundige test. Både unit-test, glassbox-test(whitebox-test), og blackbox-test, som krævet i opgaveformuleringen. Desuden har vi tilføjet en mindre bruger-accepttest. Programmet virker efter hensigten. Opskrifterne konverteres som de skal, og det er nemt at redigere måleenhederne og tilføje nye. Alt dette er fyldestgørende dokumenteret.

Det har været en spændende at benytte de værktøjer vi har lært om i kurset. De viser sig at være meget anvendelige i praksis. Vi kunne nemt, ud fra opgaveformuleringen, få fast lagt kravspecifikationen. Med denne klar, var også designdelens første iteration hurtigt overstået. Nu var det ligetil at fastlægge interfacet mellem klasserne, og på den måde opdele arbejdet i flere sideløbende processer. Det er her, det overblik de grundige indledende faser gav os, viser sin styrke. Arbejdet med både at kode og teste kom hurtigt fra hånden, og vi nåede på den måde gennem tre iterationer uden at bryde vores tidsplan.

En præcis kravspecifikation, en grundig analyse og et detaljeret design gør det nemt at lave en realistisk tidsplan. Det effektiviserer også arbejdet, da det bliver klart hvilke processer, der kan forgå samtidigt. F.eks. kunne vi, da designfasen var overstået, dele arbejdet mellem kodeskrivning og testplanlægning, fordi vi havde fået defineret de forskellige klasser og deres interface klart og tydeligt.

Det blev også tydeligt for os, hvor viktig testning er. Selvom det er et ret lille og ukompliceret program vi har lavet, fandt vi fejl, som vi ikke ville have fundet bare ved brug af sund fornuft og den afprøvning, der anvendes når man koder. Vi opdagede dem først, da der blev testet grundigt og systematisk. Men det er også tydeligt at testning er en tids- og mandskabsforbrugende del af et projekt, og hvis man føler sig på sikker grund, kan det være fristende at slække på indsatsen på denne del.

Hvis vi skal pege på noget, der kunne laves anderledes ved programmet, er det de filformater programmet tager. Som det er nu tager det bare *.txt, og det vil være fint hvis det umiddelbart kan tage eksempelvis *.doc, *.pdf, *.html, etc. uden de skal omkring udklipsholderen. Vi mente dog, at vi ikke ville bruge tid på dette. Programmet

fungerer fint i den nuværende form, og mængden af ekstra kode- og testarbejde dette ville medføre, gav ikke mening i forhold til opgavens indhold.

En bemærkning om måleenhedene. Som programmet er nu, er mængden af mål, der er registreret begrænset til de mest almindelige, og til et udpluk af disse forkortelser. Desuden kan det diskuteres i lang tid (som det blev gjort i gruppen), hvordan de forskellige enheder bør opgives. Programmet lægger op til at brugeren selv kan redigere målene, så vi har derfor valgt ikke at gøre mere ud af dette. Filen med måle enheder er også skilt ud fra selve kildekoden, og er derfor egentlig ikke et programmeringspørgsmål længere. Ændringer i programmet har som sådan ingen indflydelse på måleenhederne, og ændringer i måleenhederne ikke på programmet.

Alt i alt har det været en lærerig og spændende proces, og alle i gruppen føler, at vi har fået nogle værktøjer, som vil være meget anvendelige, både i vores videre studie og senere.

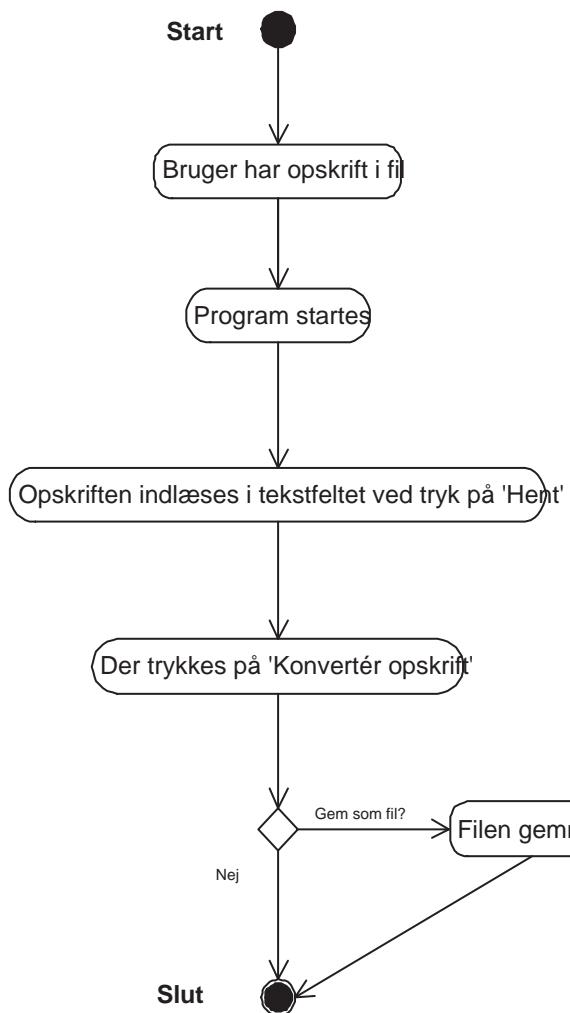
Litteratur

- [1] Råd og vejledning om projektarbejde og rapportskrivning, *Layla Dybkjær*, 2004
- [2] Slides fra undervisningen fra:
<http://www.nis.sdu.dk/education/datalogi/index.php?include=NIS02>
- [3] Softwaretest - kom godt i gang, *Klaus Olsen og Poul staal vinje*, 2004, 1. udgave
- [4] Software Engineering, *Ian Sommerville*, 7.th edt Addison Wesley 2004

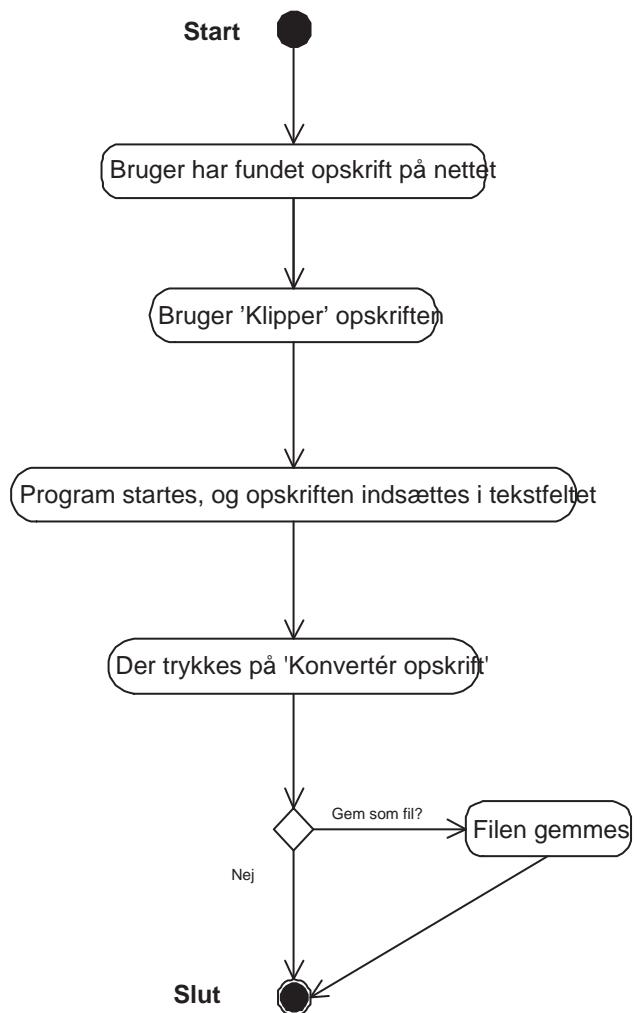
Bilag A

UML diagrammer

UML - Aktivitets Diagram

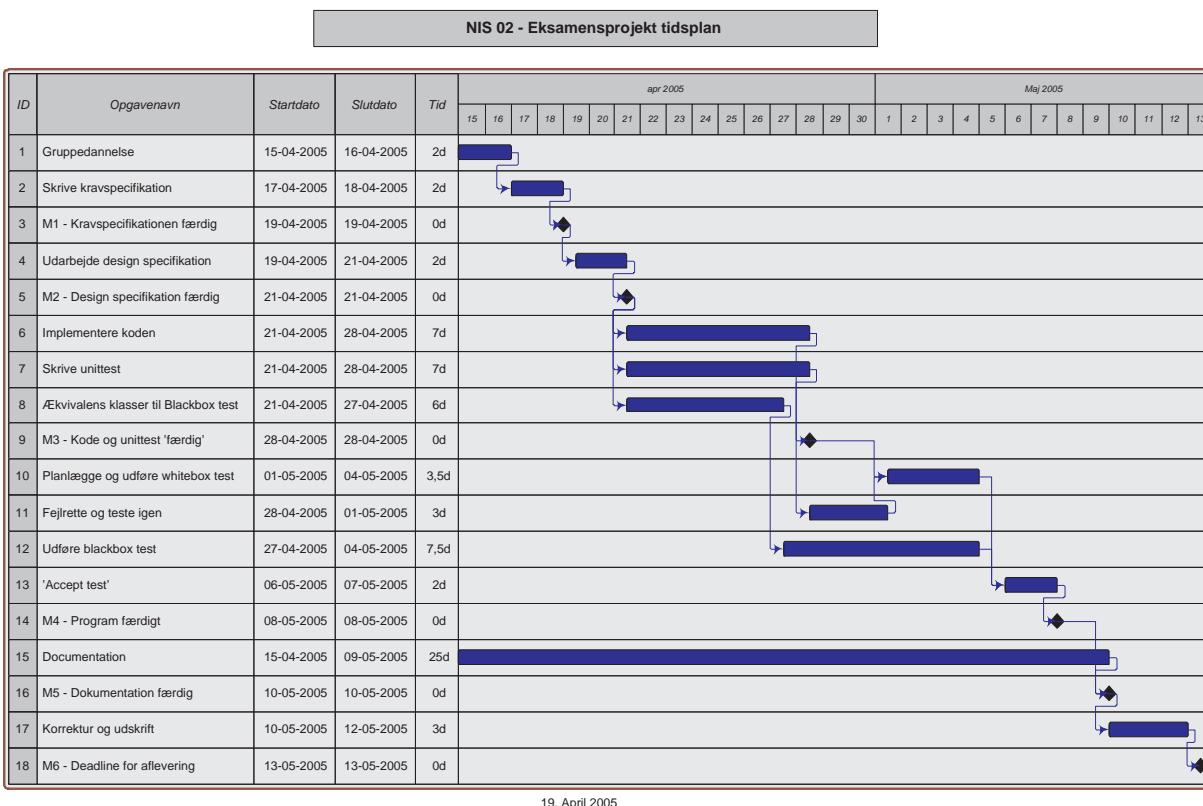


UML - Aktivitets Diagram



Bilag B

Tidsplan



Bilag C

Møde referater

Referat af gruppemøde 19. april 05 NIS 02 - Eksamensprojekt - Møde nummer 1.

1. Officiel gruppedannelse: Kristian, Anders, Jogeir og Kokken, med en målsætning som tidligere i den bedre ende af skalaen (Så godt vi kan, indenfor rimelighedens grænse)
2. Referent: Kokken
3. Næste møde Torsdag d. 21/4 kl. 12.15
4. Tidsplan med milepæle udarbejdet.
5. Udviklingsmodel: Iterativ metode, når kravspecifikationen er lavet
6. Design note: i. JAVA er programmeringssproget (version 5.0) ii. Eclipse er editor/compiler iii. CVS benyttes
7. Arbejdsopgaver og rollefordeling lægges fast i forbindelse med designspecification
8. Risikovurdering i. Sygdom lav/medium ii. Hardwarefejl meget lav iii. Softwarefejl lav iv. Fejl i vurdering af tidsplan høj risiko v. M1 kravspecifikation meget lav vi. M2 Design specifikation færdig lav vii. M3 Kode og Unit test færdig middel viii. M4 program/testfase færdig høj ix. M5 documentation færdig høj x. M6 Deadline for aflevering meget lav
9. Kage til næste møde: Kokken

Referat fra gruppemøde 21.april 2005 NIS02 Eksamensprojekt møde nummer 2

1. Kravspecifikation klar
 - Læst igennem

- Efter møde med opgavestiller Laila Dybkjær er kravspecifikation ændret. Specifikation er ændret iht. krav om at der skal være nemt for brugeren at tilføje nye mål.
2. Tidsplan 1.udkast er klar.
 3. Diskuterer design af program
 - klasse struktur
 - Besluttet at bruge GUI
 - Besluttet hvilke metoder/ funktioner der skal bruges
 4. Påmindelse om, at der skal laves brugervejledning
 5. Fordeling af arbejdsopgaver
 - Designspecifikation + klasserne converter, filereader/ writer (Kristian)
 - Kravspecifikation justeres + GUI klasser (Anders)
 - Brugsmønstre + konverteringsliste og junittest (Jacob)
 - Blackboxtest + junittest og en liste over hvad kan gå galt (Jogeir)
 6. Opgaver fra første møde er opfyld iht. Tidsplan, ingen ændring
 7. Beslut at alle dokumenter skrives i .txt og det hele samles i Latex.
 8. Næste møde er mandag 25 april fra kl. 10.15

Referat af gruppemøde 25/4 - 05 NIS 02 - Eksamensprojekt - møde nummer 3

1. Status:
 - Koden er vel undervejs, enkelte småting mangler
 - CVS, er opsat og fungerer
2. Til rapportering:
 - Tegnsæt observation, UTF-8 anvendes frem for ASCII, på grund af brøktal ikke eksisterer i ASCII (UTF-8 er valgt som ISO standart)
3. Save/output format manler
4. Tidsplan: Vi forventer at nå næste milepæl
5. Ting vi skal håndtere: Gasmark 5.

Referat af gruppemøde 28. april 05 NIS 02 - Eksamensprojekt - Møde nummer 4.

1. Tidsplan: Holder, vi justerer dog forventningerne lidt, mht. fejljustering og yderligere software test
2. Design/Kravspecifikation:
 - Rettes tilmed følgende: Slettefunktion til omregningsenheder
 - Typecheck på editeringsfelter
3. Junit og blackbox test udført, følgende debatteret og besluttet
 - Linieskift indsættes efter filindlæsning = Dokumenteres væk, under bagatelgrænsen
 - Problemer med at håndtere store bogstaver = Fejl skal rettes
 - Fejl med tabulering der erstattes af mellemrum = Fejl skal rettes
 - Regnfejl: $2,54 * 2,75 = 6,9849997$ (skulle være 6,985) = Overflow error, Problemet opstår med forkert præcision ved at bruge Double i stedet for float, dokumentation skrives om problemet og fejlen rettes
 - Fejl med at gemme linieskift (unix/win) = fejl skal rettes
 - Manglende fejlmeddeelse ved tom fil og ikke .txt fil = fejl skal rettes
 - Starter med tal, overses af genkenderen = fejl skal rettes
 - Problemer, ved komma i stedet for punktum = ignoreres, idet komma som oftest bruges som tusind-separator
 - $1 \frac{1}{2}$ omregnes forkert, når der er mellemrum = fejlen skal rettes
 - Problem med bindestreger = fejl rettes
 - Tal skrevet som bogstavstrenge (fx "twentyone") = Ignoreres, dokumentation skrives

Referat af gruppemøde 2. maj 05 NIS 02 - Eksamensprojekt - Møde nummer 5.

1. Afbud: Anders (syg)
2. Tidsplan:
 - Holder, vi justerer dog arbejdsopgaverne lidt, enkelte opgaver omfordeles fra programmeringen pga. sygdom
3. Design:
 - Afrunding: Java benytter afrundingsformen 'Half even', der er minimalt forskelligt fra almindelig dansk afrundingsregler. Dette dokumenteres og rettes ikke

- Hvis enheder forkortes, og står først (fx in. 3) skal dette korrigeres af brugeren ved at indtaste in. I listen over omregninger
4. Test:
- Whitebox tesning foretages intensivt af RecepConverter klassen, idet den vigtigste del af funktionaliteten ligger heri
5. Arbejdet går planmæssigt frem, og mødet sluttede i god ro og orden

Bilag D

Test opskrifter

D.1 Lord Baltimore Cake

INGREDIENTS: 2 3/4 cups cake flour 1 tablespoon baking powder 1 1/4 teaspoons salt
2/3 cup butter, softened 2 cups white sugar 7 egg yolks 1 1/4 cups milk 2 teaspoons
vanilla extract 1/2 cup crushed macaroon cookies 1/2 cup chopped pecans 1/4 cup
chopped blanched almonds 12 candied cherries, quartered 2 teaspoons lemon juice 1
teaspoon orange zest 1 recipe Seven Minute Frosting

Preheat oven to 350 degrees F (175 degrees C). Grease and flour three 9 inch round layer cake pans. In small bowl combine flour, baking powder and salt. In another small mixer bowl beat egg yolks until thick and lemon colored; set aside. In large mixer bowl combine butter and sugar. Beat until very light and fluffy, scraping bowl occasionally. Beat in egg yolks. With mixer at low speed add dry ingredients alternately with milk and vanilla, starting and ending with dry ingredients. Divide evenly among prepared pans. Spread to edges. Bake at 350 degrees F (175 degrees C) for 20 to 25 minutes, until golden. Let cool in pans for 10 minutes. Turn out onto wire racks to cool completely.

To Make Filling: Make Seven Minute Frosting. Remove one third of the Seven Minute Frosting and place it in a mixing bowl with the macaroon crumbs, pecans, almonds, cherries, lemon juice, and orange rind. Fold together until thoroughly blended. Use this mixture as the filling between the three cake layers, and use the remaining 2/3 of the frosting to cover the tops and sides of the cake.

D.2 Maraschino Cherry Nut Cake

INGREDIENTS: 2 1/4 cups cake flour 2 1/2 teaspoons baking powder 1/2 teaspoon salt
1/2 cup shortening 1 1/3 cups white sugar 3 egg whites 2/3 cup milk 1 (10 ounce)
jar maraschino cherries 1/2 cup chopped pecans 3/4 cup butter 6 cups confectioners'
sugar 1/3 cup milk 6 drops red food coloring 1 1/2 teaspoons vanilla extract 1 (4 ounce)
jar maraschino cherries

DIRECTIONS: Preheat oven to 350 degrees F (175 degrees C). Grease and lightly flour two 8 or 9 inch round cake pans or one 9x13 inch cake pan. Reserve 1/4 cup maraschino cherry juice. Coarsely chop the cherries to make 1/2 cup. Set aside. Combine flour, baking powder, and 1/4 teaspoon of the salt in a small bowl and set aside. Beat shortening in a large bowl with an electric mixer on medium high speed for 30 seconds. Add the 1 1/3 cups white sugar and beat until well combined. Add the egg whites, one at a time, beating well after each. Combine 2/3 cup milk and 1/4 cup cherry juice. Add the flour and milk mixture alternately to the shortening mixture, beating on low speed after each addition until just combined. Stir in the chopped cherries and nuts. Pour batter into prepared pans. Bake in a 350 degrees F (175 degrees C) for 25 to 30 minutes for two 8 or 9 inch round cakes or for 30 to 35 minutes for a 9x13 inch pan. Cool cakes in pans on a wire rack for 10 minutes, remove from pans and allow to them to cool fully before frosting. To Make Butter Frosting: Beat 3/4 cups butter in a large bowl till fluffy. Gradually add 3 cups sifted confectioners' sugar, beat well. Slowly beat in 1/3 cup milk, 1 1/2 teaspoons vanilla and 1/4 teaspoon salt. Gradually beat in the remaining 3 cups sifted confectioners' sugar. Beat in additional milk (1 to 2 tablespoons) if needed, to make frosting of spreading consistency. If desired tint the frosting pink by adding 6 drops of red food coloring. Once cake is completely cool frost with butter frosting and decorate with maraschino cherries with stems.

D.3 Large White Birthday Cake

INGREDIENTS: 3 cups sifted cake flour 4 teaspoons baking powder 3/4 teaspoon salt 1/2 cup shortening 1 1/2 cups white sugar 5 egg yolk, beaten 1 1/2 teaspoons vanilla extract 1 1/4 cups milk

DIRECTIONS: Preheat oven to 350 degrees F (175 degrees C). Grease and flour a 9 x 13 inch pan. Sift together cake flour, baking powder, and salt. In a large bowl, combine shortening, sugar, yolks, and vanilla; beat with mixer. Add sifted flour mixture alternately with milk in three parts; continue to beat until just blended. Pour batter into prepared pan. Bake for about 35 minutes. Remove from pan, and cool on a wire rack. Ice with Fluffy Boiled Icing. Sprinkle with coconut and chopped maraschino cherries if you wish.

D.4 Dirt Cake 1

This is a great conversation piece at parties. Adults love it as much as the children do. Get a new garden trowel, medium sized flower pot and artificial flower at a craft store for full effect."Original recipe yield: 1 medium size flower pot.

INGREDIENTS: 1/2 cup butter, softened 1 (8 ounce) package cream cheese, softened 1/2 cup confectioners' sugar 2 (3.5 ounce) packages instant vanilla pudding mix 3 1/2

cups milk 1 (12 ounce) container frozen whipped topping, thawed 32 ounces chocolate sandwich cookies with creme filling

DIRECTIONS: Chop cookies very fine in food processor. The white cream will disappear. Mix butter, cream cheese, and sugar in bowl. In a large bowl mix milk, pudding and whipped topping together. Combine pudding mixture and cream mixture together. Layer in flower pot, starting with cookies then cream mixture. Repeat layers. Chill until ready to serve. Add artificial flower and trowel. Enjoy!

Bilag E

Kildekode

E.1 MainGUI.java

```
1
2 import java.awt.BorderLayout;
3 import java.awt.Dimension;
4 import java.awt.FlowLayout;
5 import java.awt.Toolkit;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.io.File;
9
10 import javax.swing.JFileChooser;
11 import javax.swing.JFrame;
12 import javax.swing.JMenu;
13 import javax.swing.JMenuBar;
14 import javax.swing.JMenuItem;
15 import javax.swing.JOptionPane;
16 import javax.swing.JPanel;
17 import javax.swing.JScrollPane;
18 import javax.swing.JTextArea;
19 import javax.swing.border.EmptyBorder;
20 import javax.swing.JButton;
21
22 /**
23 * Denne klasse udgør selve hovedvinduet i RecipeConverter
24 *
25 * @author Kristian Kræmmer Nielsen <jkkn@jkkn.dk>
26 * @author Jøgeir Anderssen <jogeir.anderssen@gmail.com>
27 * @author Jacob Mikkelsen <kokken@grydeske.dk>
28 * @author Anders Brysting <andersbrysting@nal-net.dk>
29 * @version 1.0
```

```
30  * Created on 2005-04-21
31  */
32 public class MainGUI {
33     private JFrame frame;
34     private JTextArea recipeText;
35     private JPanel buttonPane;
36     private JPanel textPane;
37
38     /** Denne funktion åbner GUI vinduet
39      */
40     private void openGUI() {
41         //The frame itself
42         frame = new JFrame("Konvertér_Kage_Opskrifter");
43         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
44
45         // Holding the content of the frame
46         JPanel contentPane = (JPanel) frame.getContentPane();
47         contentPane.setBorder(new EmptyBorder(6, 6, 6, 6));
48
49         // Setting the overall layout
50         contentPane.setLayout(new BorderLayout(6, 6));
51
52         // Panel that hold the text textfield for the recipe.
53         textPane = new JPanel();
54         textPane.setBorder(new EmptyBorder(6, 6, 6, 6));
55         textPane.setLayout(new BorderLayout(6, 6));
56
57         // The textfield for the recipes to convert.
58         recipeText = new JTextArea();
59             recipeText.setFocusable(true);
60             recipeText.setLineWrap(true);
61             recipeText.setWrapStyleWord(true);
62             JScrollPane scrollActionTextPane = new JScrollPane(
63                 recipeText);
64             textPane.add(scrollActionTextPane, BorderLayout.CENTER);
65
66         // The panel holding the buttons to interact with the program
67
68         buttonPane = new JPanel();
69         buttonPane.setLayout(new FlowLayout());
70
71             JButton get = new JButton("Hent_Opskrift");
72             get.addActionListener(new ActionListener() {
73                 public void actionPerformed(ActionEvent e)
```

```
73          {
74              inputFile();
75          }
76      });
77      buttonPane.add(get);
78
79      JButton save = new JButton("Gem\u00f8pskrift");
80      save.addActionListener(new ActionListener() {
81          public void actionPerformed(ActionEvent e)
82          {
83              outputFile();
84          }
85      });
86      buttonPane.add(save);
87
88      JButton convert = new JButton("Konvert\u00e9r");
89      convert.addActionListener(new ActionListener() {
90          public void actionPerformed(ActionEvent e)
91          {
92              convert();
93          }
94      });
95      buttonPane.add(convert);
96
97      JButton clear = new JButton("Ryd");
98      clear.addActionListener(new ActionListener() {
99          public void actionPerformed(ActionEvent e)
100         {
101             clearTextArea();
102         }
103     });
104     buttonPane.add(clear);
105
106     JButton editMess = new JButton("Redig\u00e9r\u00e5m\u00e5l");
107     editMess.addActionListener(new ActionListener() {
108         public void actionPerformed(ActionEvent e)
109         {
110             editMeasures();
111         }
112     });
113     buttonPane.add(editMess);
114
115     contentPane.add(textPane, BorderLayout.CENTER);
116
117     contentPane.add(buttonPane, BorderLayout.SOUTH);
```

```
118
119     makeMenuBar() ;
120
121     frame . setResizable ( true ) ;
122     frame . pack () ;
123     frame . setSize ( 600 , 600 ) ;
124
125     //Center the application on the screen .
126     Dimension d = Toolkit . getDefaultToolkit () . getScreenSize () ;
127     frame . setLocation ( d . width / 2 - frame . getWidth () / 2 , d .
128                         height / 2
129                         - frame . getHeight () / 2 ) ;
130
131     frame . setVisible ( true ) ;
132 }
133 /**
134 * Creates the menus for the game
135 */
136 private void makeMenuBar()
137 {
138     JMenuBar menuBar = new JMenuBar () ;
139     frame . setJMenuBar ( menuBar ) ;
140
141     JMenu fileMenu = new JMenu ("Opskrift") ;
142     menuBar . add ( fileMenu ) ;
143
144     JMenu helpMenu = new JMenu ("Hjælp") ;
145     menuBar . add ( helpMenu ) ;
146
147     JMenuItem saveItem = new JMenuItem ("Gem_Opskrift") ;
148     saveItem . addActionListener ( new ActionListener () {
149         public void actionPerformed ( ActionEvent e )
150         {
151             outputFile () ;
152         }
153     } );
154     fileMenu . add ( saveItem ) ;
155
156     JMenuItem loadItem = new JMenuItem ("Hent_Opskrift") ;
157     loadItem . addActionListener ( new ActionListener () {
158         public void actionPerformed ( ActionEvent e )
159         {
160             inputFile () ;
161         }
162     } );
```

```
162     });
163     fileMenu.add(loadItem);
164
165     JMenuItem convrtItem = new JMenuItem("Konvertér „
166                                         Opskrift");
167     convrtItem.addActionListener(new ActionListener() {
168         {
169             public void actionPerformed(ActionEvent e)
170             {
171                 convert();
172             }
173         );
174     fileMenu.add(convrtItem);
175
176     JMenuItem editItem = new JMenuItem("Redigér „
177                                         målenheder");
178     editItem.addActionListener(new ActionListener() {
179         {
180             public void actionPerformed(ActionEvent e)
181             {
182                 editMeasures();
183             }
184         );
185     fileMenu.add(editItem);
186
187     JMenuItem quitItem = new JMenuItem("Quit");
188     quitItem.addActionListener(new ActionListener() {
189         {
190             public void actionPerformed(ActionEvent e)
191             {
192                 quit();
193             }
194         );
195     fileMenu.add(quitItem);
196
197     JMenuItem helpItem = new JMenuItem("Hvordan „
198                                         programmet benyttes");
199     helpItem.addActionListener(new ActionListener() {
200         {
201             public void actionPerformed(ActionEvent e)
202             {
203                 openHelp();
204             }
205         );
206     helpMenu.add(helpItem);
207
208     JMenuItem aboutItem = new JMenuItem("Om „opskrift „
209                                         konverteringsprogrammet");
210     aboutItem.addActionListener(new ActionListener() {
```

```

203         public void actionPerformed(ActionEvent e)
204         {
205             openAbout();
206         }
207     });
208     helpMenu.add(aboutItem);
209 }
210
211
212
213 /**
214 * Denne funktion håndterer at brugeren klikker på "Hent
215 * Opskrift"
216 * Opskriften vises i det store tekst vindu.
217 */
218 private void inputFile() {
219     JFileChooser chooser = new JFileChooser();
220     chooser.setCurrentDirectory(new File("texts/"));
221     int returnVal = chooser.showOpenDialog(frame);
222     if (returnVal == JFileChooser.APPROVE_OPTION) {
223         File file = chooser.getSelectedFile();
224         recipeText.setText(FileIO.readFile(file));
225         recipeText.setCaretPosition(0);
226     }
227 }
228
229 /**
230 * Denne funktion håndterer at brugeren klikker på "Gem Opskrift
231 * "
232 * Åbner en traditionel "gem-dialog".
233 */
234 private void outputFile() {
235     JFileChooser chooser = new JFileChooser();
236     chooser.setCurrentDirectory(new File("texts/"));
237     int returnVal = chooser.showSaveDialog(frame);
238     if (returnVal == JFileChooser.APPROVE_OPTION) {
239         File selected = chooser.getSelectedFile();
240         boolean save = false;
241         if (selected.exists()) {
242             if (JOptionPane.showConfirmDialog(frame,
243                 "Er du sikker på at du vil overskrive '" +
244                 + chooser.getSelectedFile() + "?",
245                 "Save game", JOptionPane.WARNING_MESSAGE,
246                 JOptionPane.YES_NO_OPTION) == JOptionPane.
247                     YES_OPTION) {
248                 save = true;
249             }
250         }
251     }
252 }
```

```

245         }
246     else {
247         save = true;
248     }
249     if (save) {
250         FileIO.writeFile(selected, recipeText.getText());
251     }
252 }
253 }
254
255 /**
256 * Denne funktion håndterer at brugeren klikker på "Konverter
257 * opskrift"
258 * Målene i opskriften konverteres og opskriften vises med de
259 * nye mål i tekstfeltet.
260 */
261 private void convert() {
262     recipeText.setText(RecipeConverter.convert(recipeText
263         .getText()));
264     recipeText.setCaretPosition(0);
265 }
266
267 /**
268 * Denne funktion håndterer at brugeren klikker på "Redigér
269 * målenheder"
270 * Åbner redigerings GUI'en.
271 */
272 private void editMeasures() {
273     MeasurementGUI measurementGUI = new MeasurementGUI();
274 }
275
276 /**
277 * Åbner brugervejledningen
278 */
279 private void openHelp() {
280     String helpText = FileIO.readFile(new File("texts/
281         helpFile.txt"));
282     JOptionPane.showMessageDialog(frame, helpText, "Help",
283         JOptionPane.INFORMATION_MESSAGE);
284 }
285
286 /**
287 * Åbner "Om".
288 */
289 private void openAbout() {
290     String aboutText = FileIO.readFile(new File("texts/
291         aboutText.txt"));
292     JOptionPane.showMessageDialog(frame, aboutText, "
293         About");
294 }

```

```

283         }
284
285         /**
286          * Quit function: quit the application.
287          */
288         private void quit()
289         {
290             System.exit(0);
291         }
292
293         /**
294          * nulstiller tekstfeltet.
295          */
296         private void clearTextArea(){
297             recipeText.setText("");
298         }
299
300         /**
301          * Hovedfunktionen for programmet
302          * Indeholder MainGUI's konstruktur.
303          * @param args
304          */
305         public static void main(String[] args) {
306             MainGUI mainGUI = new MainGUI();
307             mainGUI.openGUI();
308         }
309     }

```

E.2 MeasurementGUI.java

```

1
2 import java.awt.BorderLayout;
3 import java.awt.Dimension;
4 import java.awt.FlowLayout;
5 import java.awt.GridLayout;
6 import java.awt.Toolkit;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
9 import java.io.File;
10
11 import javax.swing.JFrame;
12 import javax.swing.JOptionPane;
13 import javax.swing.JPanel;
14 import javax.swing.JLabel;

```

```
15 import javax.swing.JScrollPane;
16 import javax.swing.JTextArea;
17 import javax.swing.JTextField;
18 import javax.swing.border.EmptyBorder;
19 import javax.swing.JButton;
20
21
22 /**
23  * Denne klasse præsenterer tilgængelige alle konvertingsmetoder i
24  * systemet
25  * i et vindue og tilbyder brugeren at kunne tilføje en ny
26  * konvertingsmetode.
27  *
28  * @author Kristian Kræmmer Nielsen <jkkn@jkkn.dk>
29  * @author Jogeir Anderssen <jogeir.anderssen@gmail.com>
30  * @author Jacob Mikkelsen <kokken@grydeske.dk>
31  * @author Anders Brysting <andersbrysting@nal-net.dk>
32  * @version 1.0
33  * Created on 2005-04-21
34  */
35 public class MeasurementGUI {
36     private JFrame frame;
37     private JPanel contentPane;
38     private JPanel inputPanel;
39     private JPanel bottomPanel;
40     private JTextField fromField;
41     private JTextField faktorField;
42     private JTextField toField;
43     private JTextField constField;
44     private JLabel from;
45     private JLabel faktor;
46     private JLabel to;
47     private JLabel consta;
48     private JButton add;
49     private JButton delete;
50     private JButton exit;
51     private JButton reset;
52     private JPanel textPane;
53     private JTextArea measureText;
54     private MeasurementList ml;
55
56     /*
57      * Konstruktor for editerings interfacet.
58      */
59     public MeasurementGUI() {
```

```
58             ml = new MeasurementList();
59             openGUI();
60         }
61
62     /**
63      * Åbner redigeringsvinduet.
64      */
65     public void openGUI() {
66
67         frame = new JFrame("Tilføj ny måleenhed");
68         frame.setDefaultCloseOperation(JFrame.
69                         DISPOSE_ON_CLOSE);
70
71         contentPane = (JPanel) frame.getContentPane();
72         contentPane.setBorder(new EmptyBorder(6, 6, 6, 6));
73         contentPane.setLayout(new BorderLayout(6, 6));
74
75         inputPanel = new JPanel();
76         inputPanel.setBorder(new EmptyBorder(6, 6, 6, 6));
77         inputPanel.setLayout(new GridLayout());
78
79         bottomPanel = new JPanel();
80         bottomPanel.setBorder(new EmptyBorder(0, 0, 0, 0));
81         bottomPanel.setLayout(new FlowLayout());
82
83         fromField = new JTextField();
84         faktorField = new JTextField();
85         toField = new JTextField();
86         constField = new JTextField("0");
87
88         from = new JLabel("Fra:");
89         from.setBorder(new EmptyBorder(3, 5, 3, 5));
90         faktor = new JLabel("Faktor:");
91         faktor.setBorder(new EmptyBorder(3, 5, 3, 5));
92         to = new JLabel("Til:");
93         to.setBorder(new EmptyBorder(3, 5, 3, 5));
94         consta = new JLabel("Konstant:");
95         consta.setBorder(new EmptyBorder(3, 5, 3, 5));
96
97         add = new JButton("Tilføj");
98         add.addActionListener(new ActionListener() {
99             {
100                 public void actionPerformed(ActionEvent e)
101                 {
102                     handleAdd();
103                 }
104             }
105         });
106     }
```

```
102     });
103
104     delete = new JButton("Slet");
105     delete.addActionListener(new ActionListener() {
106         public void actionPerformed(ActionEvent e) {
107             {
108                 handleDelete();
109             }
110         }
111     });
112
113     reset = new JButton("Nulstil_felter");
114     reset.addActionListener(new ActionListener() {
115         public void actionPerformed(ActionEvent e) {
116             {
117                 reset();
118             }
119         }
120
121         exit = new JButton("Luk_vinduet");
122         exit.addActionListener(new ActionListener() {
123             public void actionPerformed(ActionEvent e) {
124                 {
125                     exit();
126                 }
127             }
128         });
129         inputPanel.add(from);
130         inputPanel.add(fromField);
131
132         inputPanel.add(faktor);
133         inputPanel.add(faktorField);
134
135         inputPanel.add(to);
136         inputPanel.add(toField);
137
138         inputPanel.add(consta);
139         inputPanel.add(constField);
140
141         bottomPanel.add(add);
142         bottomPanel.add(delete);
143         bottomPanel.add(reset);
144         bottomPanel.add(exit);
145
146         textPane = new JPanel();
147         textPane.setBorder(new EmptyBorder(3, 6, 6, 6));
```

```
147         textPane.setLayout(new BorderLayout(6, 6));
148
149         measureText = new JTextArea();
150         measureText.setEditable(false);
151         JScrollPane scrollActionTextPane = new JScrollPane(
152             measureText);
153         textPane.add(scrollActionTextPane, BorderLayout.
154             CENTER);
155
156         contentPane.add(inputPanel, BorderLayout.NORTH);
157         contentPane.add(textPane, BorderLayout.CENTER);
158         contentPane.add(bottomPanel, BorderLayout.SOUTH);
159
160         frame.pack();
161         frame.setResizable(false);
162         frame.setSize(600,600);
163         frame.setMinimumSize(new Dimension(600, 600));
164
165         Dimension d = Toolkit.getDefaultToolkit().
166             getScreenSize();
167         frame.setLocation(d.width / 2 - frame.getWidth() / 2,
168             d.height / 2
169             - frame.getHeight() / 2);
170
171         measureText.setText(getMeasures());
172
173         frame.setVisible(true);
174     }
175
176     /**
177      * henter en streng indeholde de kendte mål.
178      * @return Streng indeholdende kendt måleenheder og deres
179      * konverterings metoder
180     */
181     private String getMeasures() {
182
183         return ml.toString();
184     }
185
186     /**
187      * Håndterer at brugeren klikker på "tilføj"
188      */
189     private void handleAdd() {
190         //Håndterer tomme tekstmærker
191         Boolean emptyFields = false;
```

```

187     try {
188         String fromText = fromField.getText();
189         String toText = toField.getText();
190         String fakt = faktorField.getText();
191         String konst = constField.getText();
192         if (fromText.equals("") || toText.equals("") || fakt.equals(""))
193             || konst.equals("")) {
194             emptyFields = true;
195         }
196         Double faktor = Double.parseDouble(faktorField.getText())
197             ;
198         int konstant = Integer.parseInt(constField.getText());
199     }
200     catch (Exception e) {
201         emptyFields = true;
202     }
203     if (!emptyFields) {
204         Measurement m = new Measurement(fromField.getText(),
205             toField.getText(), Double.parseDouble(faktorField.
206                 getText()), Integer.parseInt(constField.getText()));
207         if (ml.contains(fromField.getText())){
208             String overwriteText = "Ønsker du at overskrive " +
209                 fromField.getText();
210             int del = JOptionPane.showConfirmDialog(frame,
211                 overwriteText, "Overskrivning", JOptionPane.
212                 YES_NO_OPTION);
213             if (del == JOptionPane.YES_OPTION) {
214                 ml.addMeasurement(m);
215             }
216             } else {
217                 ml.addMeasurement(m);
218             }
219             measureText.setText(getMeasures());
220         } else {
221             String fejlText = FileIO.readFile(new File("texts/
222                 fejlText.txt"));
223             JOptionPane.showMessageDialog(frame, fejlText, "Fejl",
224                 JOptionPane.INFORMATION_MESSAGE);
225         }
226         reset();
227     }
228
229
230 /**
231 * Lukker vinduet, når der klikkes på "luk vinduet".

```

```

223         */
224     private void exit() {
225         frame.dispose();
226     }
227
228     /**
229      * Håndterer at brugeren klikker på "slet"
230      */
231     private void handleDelete() {
232         String deleteText = JOptionPane.showInputDialog(frame
233             , "Skriv navnet på den enhed der skal slettes.");
234         if (ml.getMeasurement(deleteText) != null) {
235             String delText = "Ønsker du at slette " +
236                 deleteText;
237             int del = JOptionPane.showConfirmDialog(frame
238                 , delText, "Slet", JOptionPane.YES_NO_OPTION)
239                 ;
240             if (del == JOptionPane.YES_OPTION) {
241                 ml.removeMeasurement(deleteText);
242                 measureText.setText(getMeasures());
243             }
244         }
245
246     /**
247      * nulstiller inputfeltene der bruges til redigering.
248      */
249     private void reset(){
250         fromField.setText("");
251         faktorField.setText("");
252         toField.setText("");
253         constField.setText("0");
254     }
255 }
```

E.3 FileIO.java

```

1
2 import java.io.BufferedReader;
```

```
3 import java.io.File;
4 import java.io.FileOutputStream;
5 import java.io.IOException;
6 import java.io.OutputStreamWriter;
7 import java.io.StringWriter;
8 import java.util.Scanner;
9
10 import javax.swing.JFrame;
11 import javax.swing.JOptionPane;
12
13 /**
14  * Denne klasse håndterer alt fil ind og output.
15  *
16  * @author Kristian Kræmmer Nielsen <jkkn@jkkn.dk>
17  * @author Jøgeir Anderssen <jogeir.anderssen@gmail.com>
18  * @author Jacob Mikkelsen <kokken@grydeske.dk>
19  * @author Anders Brysting <andersbrysting@nal-net.dk>
20  * @version 1.0
21  * Created on 2005-04-21
22  */
23 public class FileIO {
24
25     /**
26      * @param text Den text-fil der skal hentes
27      * @return teksten fra filen i form af en streng.
28      */
29     public static String readFile(File text) {
30         StringWriter string = new StringWriter();
31         BufferedWriter buffer = new BufferedWriter(string);
32         try {
33             Scanner file = new Scanner(text, "UTF-8");
34             while (file.hasNext()) {
35                 buffer.append(file.nextLine());
36                 if (file.hasNext()) buffer.newLine();
37             }
38             buffer.flush();
39         } catch (IOException ioe) { ioe.printStackTrace(); }
40         if (string.toString().equals("")) {
41             JFrame frame = new JFrame("");
42             String fejlText = "Der er sket en fejl med "
43                         + "filindlæsningen, \n nenten er filen tom, eller den er "
44                         + "ikke i .txt format";
45             JOptionPane.showMessageDialog(frame, fejlText, "Fejl",
46                                         JOptionPane.INFORMATION_MESSAGE);
47         }
48     }
49 }
```

```

45         }
46             return string.toString();
47         }
48
49     /** Skriver en fil
50      *
51      * @param file Filen teksten skal gemmes i.
52      * @param text tekst, der gemmes
53      */
54     public static void writeFile(File file, String text)
55     {
56         try
57     {
58         BufferedWriter out = new BufferedWriter(
59             new OutputStreamWriter(new FileOutputStream(file)
60             , "UTF-8"));
61         out.write(text);
62         out.close();
63     }
64     catch(IOException e)
65     {
66         System.out.println(e);
67     }
68 }
69 }
```

E.4 Measurement.java

```

1 import java.text.DecimalFormat;
2 import java.util.regex.Matcher;
3 import java.util.regex.Pattern;
4
5 /**
6  * Denne klasse udgør en konverteringsmetode fra en enhed til en
7  * anden
8  * @author Kristian Kræmmer Nielsen <jkkn@jkkn.dk>
9  * @author Jøgeir Anderssen <jogeir.anderssen@gmail.com>
10 * @author Jacob Mikkelsen <kokken@grydeske.dk>
11 * @author Anders Brysting <andersbrysting@nal-net.dk>
12 * @version 1.0
13 * Created on 2005-04-21
14 */
15 public class Measurement {
```

```

16
17     public static final String NUMBERSIGNS = "\u00a4\u00a5\u00a6\u00a7\u00a8\u00a9";
18     private static final DecimalFormat FORMAT = new DecimalFormat(
19         "#,##0.##");
20     private static final Pattern DIVIDERS = Pattern.compile("\u00a4*\u00a5*\u00a6*\u00a7*\u00a8*\u00a9*");
21     private String from;
22     private String to;
23     private double factor;
24     private int shiftFactor;
25
26     /**
27      * @param from Navn på fra enhed
28      * @param to Navn på til enhed
29      * @param factor Faktor, som ganges på fra-enhed for at opnå til-
30      *               enhed
31      * @param shiftFactor Forskydelse, f.eks. er Fahrenheit -32 i
32      *               forhold til Celsius
33      */
34     public Measurement(String from, String to, double factor, int
35                         shiftFactor) {
36         this.from = from;
37         this.to = to;
38         this.factor = factor;
39         this.shiftFactor = shiftFactor;
40     }
41
42     /**
43      * Opret ny konvertingsform udfra text-streng
44      * Brugt til at indlæse en measurement fra f.eks. en fil.
45      * @param input Tab-seperaret streng indeholdende fra-enhed, til-
46      *               enhed, faktor og forskydelsesfaktor
47      */
48     public Measurement(String input) {
49         String[] method = input.split("\t+");
50         this.from = method[0];
51         this.to = method[1];
52         this.factor = Double.parseDouble(method[2]);
53         this.shiftFactor = Integer.parseInt(method[3]);
54     }
55
56     /**
57      * Returner navnet på fra-enheden
58      *
59      */
60 
```

```
54     * @return navn på fra-enhed .
55     */
56     public String getFrom() {
57         return from;
58     }
59
60     /** Konvertering fra enhed til ny enhed
61     *
62     * @param number Number, der skal konverteres (uden enhed)
63     * @return Konverteret nummer afrundet til max. 2 decimaler (uden
64     * enhed)
65     */
66     public String convert(String number) {
67         return FORMAT.format((convertToNumber(number)+shiftFactor)*
68             factor);
69     }
70
71     /** Konverterer et nummer til en double
72     * Denne funktion konverterer før og fremmest brøker på formen x/
73     * y.
74     * Denne funktion håndterer også specialtegn , så som  $\frac{1}{4}$ ,  $\frac{1}{2}$  ,  $\frac{3}{4}$ 
75     * og UTF-8 karakterene: 1/3, 2/3, 1/8, 3/8, 5/8, 7/8
76     *
77     * @param number Nummer som decimaltal. (skal være UTF8 encodet)
78     * @return
79     */
80     protected static double convertToNumber(String number) {
81         double result = 0;
82         /** konvertering af brøker */
83         Matcher m = DIVIDERS.matcher(number);
84         while (m.find()) {
85             double x = Double.parseDouble(m.group(1));
86             double y = Double.parseDouble(m.group(2));
87             result += (x/y);
88         }
89         number = m.replaceAll("");
90         /** konvertering af specialtegn */
91         if (number.indexOf("1/4")!=-1) result += (1/4.);
92         if (number.indexOf("1/2")!=-1) result += (1/2.);
93         if (number.indexOf("3/4")!=-1) result += (3/4.);
94         if (number.indexOf("1/3")!=-1) result += (1/3.);
95         if (number.indexOf("2/3")!=-1) result += (2/3.);
96         if (number.indexOf("1/8")!=-1) result += (1/8.);
97         if (number.indexOf("3/8")!=-1) result += (3/8.);
98         if (number.indexOf("5/8")!=-1) result += (5/8.);
```

```

96         if (number.indexOf(0x215E)!=-1) result += (7/8.);
97         number = number.replaceAll ("["+NUMBERSIGNS+"]", "");
98         if (number.length ()>0) {
99             result += Double.parseDouble(number);
100        }
101        return result;
102    }
103
104    /** Returner enhed på slutresultatet .
105     * @return
106     */
107    public String getTo()
108    {
109        return to;
110    }
111
112    /** Returner repræsentation af omregning
113     * Returnere repræsentation af omregning , som også kan bruges til
114     * at initialisere et nyt objekt
115     */
116    public String toString() {
117        return from+"\t\t"+to+"\t\t"+factor+"\t\t"+shiftFactor;
118    }
119 }
```

E.5 MeasurementList.java

```

1
2 import java.util.HashMap;
3 import java.util.HashSet;
4 import java.util.Set;
5 import java.io.BufferedReader;
6 import java.io.File;
7 import java.io.IOException;
8 import java.io.StringWriter;
9
10 /**
11  * Denne klasse holder styr på alle tilgængelig konvertingsmetoder i
12  * systemet
13  * og har mulighed for at gemme og indlæs dem fra disk .
14  * Den tilbyder desuden en funktion til at konvertering fra én given
15  * britisk enhed
```

```
16 * @author Kristian Kræmmer Nielsen <jkkn@jkkn.dk>
17 * @author Jøgeir Anderssen <jogeir.anderssen@gmail.com>
18 * @author Jacob Mikkelsen <kokken@grydeske.dk>
19 * @author Anders Brysting <andersbrysting@nal-net.dk>
20 * @version 1.0
21 * Created on 2005-04-21
22 */
23 public class MeasurementList {
24
25     private final static File measurementsFile = new File("texts/
26             measurements.txt");
27     private HashMap<String, Measurement> methods; // med fra-
28             enhed som nøgle
29
30     /**
31      * konstruktør - indlæser liste
32      */
33     public MeasurementList() {
34         methods = new HashMap<String, Measurement>();
35         load();
36     }
37
38     /**
39      * Privat funktion til at indlæse konvertingsmetoder
40      * fra en fil.
41      */
42     private void load() {
43         String inputFile = FileIO.readFile(measurementsFile);
44         String[] lines = inputFile.split("[\n\r]+");
45         for (String line: lines) {
46             Measurement m = new Measurement(line);
47             methods.put(m.getFrom().toLowerCase(), m);
48         }
49     }
50
51     /**
52      * Gemmer konvergsmetoder til fil
53      */
54     private void save() {
55         FileIO.writeFile(measurementsFile, toString());
56     }
57
58     /**
59      * Tilføjer en ny konvertingsmetode til listen
60      * @param measurement Ny konvertingsmetode
61      */
62 }
```

```
59     public void addMeasurement(Measurement measurement) {
60         methods.put(measurement.getFrom().toLowerCase(), measurement)
61             ;
62         save();
63     }
64     /**
65      * Fjerner et mål fra map'et
66      * @param key Fra-enheden til det mål der skal fjernes.
67      */
68     public void removeMeasurement(String key){
69         methods.remove(key.toLowerCase());
70         save();
71     }
72
73     /**
74      * Returnerer en konverteringen fra angivede britiske enhed
75      *
76      * @param from Enhed, der skal konverteres fra, f. eks. "inch"
77      * @return Returnerer Measurement-konverteringsobjekt
78      */
79     public Measurement getMeasurement(String from) {
80         return methods.get(from.toLowerCase());
81     }
82
83     /**
84      * Returnerer alle britiske enheder, der kan konverteres fra
85      *
86      * @return Liste af navne på enhederne.
87      */
88     public String[] getMeasurementTypes() {
89         Set<String> types = methods.keySet();
90         return types.toArray(new String[types.size()]);
91     }
92
93     /**
94      * Returnerer alle kendte opregnings metoder
95      *
96      * @return Set af metoder
97      */
98     public Set<Measurement> getMeasurements() {
99         return new HashSet<Measurement>(methods.values());
100    }
101
102    /**
```

```

103         * Udskriver de registrerede måleenheder og deres
104             konverterings metode .
105         * En på hver linie .
106         * @return measures En streng indeholdene en tabel med kendt
107             mål .
108         */
109     public String toString(){
110         StringWriter out = new StringWriter();
111         try {
112             BufferedWriter buffer = new BufferedWriter(out);
113             for (Measurement m: methods.values()) {
114                 buffer.append(m.toString());
115                 buffer.newLine();
116             }
117             buffer.flush();
118         } catch (IOException ioe) {
119             ioe.printStackTrace();
120         }
121         return out.toString();
122     }
123     public boolean contains(String key){
124         return methods.containsKey(key);
125     }
126 }
```

E.6 RecipeConverter.java

```

1
2
3 import java.util.regex.Matcher;
4 import java.util.regex.Pattern;
5
6 /*
7  * Created on 2005-04-21
8  */
9
10 /**
11  * Denne klasse foretager en konverteringen fra en opskrift-tekst med
12  * britisk mål
13  * til en opskrift-tekst med metriske mål .
14  * Den benytter MeasurementList til alt omregning .
15  * @author Kristian Kræmmer Nielsen <jkkn@jkkn.dk>
```

```

16 * @author Jogeir Anderssen <jogeir.anderssen@gmail.com>
17 * @author Jacob Mikkelsen <kokken@grydeske.dk>
18 * @author Anders Brysting <andersbrysting@nal-net.dk>
19 * @version 1.0
20 */
21 public class RecipeConverter {
22
23     private enum PatternType {
24         NUMBER_UNIT,
25         UNIT_NUMBER,
26         NUMBER_X_NUMBER_UNIT
27     }
28
29     /** Returnerer Pattern objekt til at finde britiske enheder.
30     * Denne funktion opbygger en Regular expression mønster
31     * udfra navne på de kendte britiske enheder.
32     * Mønsteret opbygges til en streng af følgende type:
33     * "(?<=\s|^)([\d\.\frac{1}{4}\frac{1}{2}\frac{3}{4}]+)([\s°-]+)(lb|pound|in|inch|tablespoon|p
34     |pint|Fahrenheit)(?= [\s\.,;-]|$)"
35     *
36     * Her betyder:
37     *     ^ = starten af linjen
38     *     $ = slutning af linjen
39     *     \s = et mellemrum/linjeskifte, el. lignende
40     *     \d = et nummer mellem 0 og 1
41     *     [] = angiver én af de angivede tegn
42     *     + = én eller flere af forestående
43     *     \. = henviser til punktum, da "." i reg.exp. betyder
44     *          noget andet.
45     *     | = betyder "eller"
46     *     () = beder Matcher'en om at returnerer hvad der stod her
47     *          tilbage således
48     *          man kan bearbejde det.
49     *     (?<=) = angiver skal komme før, men indgår ikke i selve
50     *          resultatet
51     *     (?:) = angiver skal komme efter, men indgår ikke i selve
52     *          resultatet
53     *
54     * @param list Liste, der den skal laves udfra
55     * @param type af mønster (en af PatternType)
56     * @return returnerer mønster
57     */
58     private static Pattern getMeasurementPattern(MeasurementList list
59             , PatternType type) {
60         final String hitBefore = "(?<=[\\s\\\[|\\^)";
61
62         switch (type) {
63             case NUMBER_X_NUMBER_UNIT:
64                 hitBefore += "[\\d\\.\frac{1}{4}\\frac{1}{2}\\frac{3}{4}]+";
65                 hitBefore += "(lb|pound|in|inch|tablespoon|pint|Fahrenheit)(?= [\s\.,;-]|$)";
66                 break;
67             case NUMBER_UNIT:
68                 hitBefore += "[\\d\\.\frac{1}{4}\\frac{1}{2}\\frac{3}{4}]+";
69                 hitBefore += "(lb|pound|in|inch|tablespoon|pint|Fahrenheit)(?= [\s\.,;-]|$)";
70                 break;
71             case UNIT_NUMBER:
72                 hitBefore += "(lb|pound|in|inch|tablespoon|pint|Fahrenheit)(?= [\s\.,;-]|$)";
73                 hitBefore += "[\\d\\.\frac{1}{4}\\frac{1}{2}\\frac{3}{4}]+";
74                 break;
75         }
76
77         return Pattern.compile(hitBefore);
78     }
79
80     private static void printUsage(PrintWriter writer) {
81         writer.println("Usage: java RecipeConverter [-i|-o] file");
82         writer.println("  -i: Input file (default: input.txt)");
83         writer.println("  -o: Output file (default: output.txt)");
84     }
85
86     private static void main(String args[]) {
87         FileInputStream in;
88         FileOutputStream out;
89         PrintWriter writer;
90
91         if (args.length < 1) {
92             printUsage(System.out);
93             System.exit(1);
94         }
95
96         try {
97             in = new FileInputStream(args[0]);
98             out = new FileOutputStream(args[1]);
99             writer = new PrintWriter(out);
100            convert(in, writer);
101        } catch (FileNotFoundException e) {
102            System.out.println("File not found: " + args[0]);
103            System.exit(1);
104        } catch (IOException e) {
105            System.out.println("IO error: " + e.getMessage());
106            System.exit(1);
107        }
108    }
109
110    private static void convert(InputStream in, PrintWriter writer) {
111        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
112        String line;
113
114        while ((line = reader.readLine()) != null) {
115            if (!line.startsWith("#")) {
116                String convertedLine = convertLine(line);
117                writer.println(convertedLine);
118            }
119        }
120    }
121
122    private static String convertLine(String line) {
123        String convertedLine;
124
125        if (line.startsWith("1/")) {
126            convertedLine = convertFractionalNumber(line);
127        } else {
128            convertedLine = line;
129        }
130
131        return convertedLine;
132    }
133
134    private static String convertFractionalNumber(String line) {
135        String convertedLine;
136
137        if (line.startsWith("1/")) {
138            convertedLine = convertFractionalNumberWithLeadingZero(line);
139        } else {
140            convertedLine = convertFractionalNumberWithoutLeadingZero(line);
141        }
142
143        return convertedLine;
144    }
145
146    private static String convertFractionalNumberWithLeadingZero(String line) {
147        String convertedLine;
148
149        if (line.startsWith("1/")) {
150            convertedLine = convertFractionalNumberWithLeadingZeroAndDot(line);
151        } else {
152            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDot(line);
153        }
154
155        return convertedLine;
156    }
157
158    private static String convertFractionalNumberWithoutLeadingZero(String line) {
159        String convertedLine;
160
161        if (line.startsWith("1/")) {
162            convertedLine = convertFractionalNumberWithoutLeadingZeroAndDot(line);
163        } else {
164            convertedLine = convertFractionalNumberWithoutLeadingZeroAndNoDot(line);
165        }
166
167        return convertedLine;
168    }
169
170    private static String convertFractionalNumberWithLeadingZeroAndDot(String line) {
171        String convertedLine;
172
173        if (line.startsWith("1/")) {
174            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndNoDot(line);
175        } else {
176            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndDot(line);
177        }
178
179        return convertedLine;
180    }
181
182    private static String convertFractionalNumberWithLeadingZeroAndNoDot(String line) {
183        String convertedLine;
184
185        if (line.startsWith("1/")) {
186            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDot(line);
187        } else {
188            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndDot(line);
189        }
190
191        return convertedLine;
192    }
193
194    private static String convertFractionalNumberWithLeadingZeroAndDotAndNoDot(String line) {
195        String convertedLine;
196
197        if (line.startsWith("1/")) {
198            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndDot(line);
199        } else {
200            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndNoDot(line);
201        }
202
203        return convertedLine;
204    }
205
206    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndDot(String line) {
207        String convertedLine;
208
209        if (line.startsWith("1/")) {
210            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndDot(line);
211        } else {
212            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndNoDot(line);
213        }
214
215        return convertedLine;
216    }
217
218    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDot(String line) {
219        String convertedLine;
220
221        if (line.startsWith("1/")) {
222            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndDot(line);
223        } else {
224            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDot(line);
225        }
226
227        return convertedLine;
228    }
229
230    private static String convertFractionalNumberWithLeadingZeroAndDotAndDot(String line) {
231        String convertedLine;
232
233        if (line.startsWith("1/")) {
234            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndDotAndDot(line);
235        } else {
236            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndDotAndNoDot(line);
237        }
238
239        return convertedLine;
240    }
241
242    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndDot(String line) {
243        String convertedLine;
244
245        if (line.startsWith("1/")) {
246            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndDot(line);
247        } else {
248            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndNoDot(line);
249        }
250
251        return convertedLine;
252    }
253
254    private static String convertFractionalNumberWithLeadingZeroAndDotAndNoDot(String line) {
255        String convertedLine;
256
257        if (line.startsWith("1/")) {
258            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndDot(line);
259        } else {
260            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndNoDot(line);
261        }
262
263        return convertedLine;
264    }
265
266    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDot(String line) {
267        String convertedLine;
268
269        if (line.startsWith("1/")) {
270            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndDot(line);
271        } else {
272            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDot(line);
273        }
274
275        return convertedLine;
276    }
277
278    private static String convertFractionalNumberWithLeadingZeroAndDotAndDotAndDot(String line) {
279        String convertedLine;
280
281        if (line.startsWith("1/")) {
282            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndDotAndDotAndDot(line);
283        } else {
284            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndDotAndDotAndNoDot(line);
285        }
286
287        return convertedLine;
288    }
289
290    private static String convertFractionalNumberWithLeadingZeroAndDotAndDotAndNoDot(String line) {
291        String convertedLine;
292
293        if (line.startsWith("1/")) {
294            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndDotAndNoDotAndDot(line);
295        } else {
296            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndDotAndNoDotAndNoDot(line);
297        }
298
299        return convertedLine;
300    }
301
302    private static String convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndDot(String line) {
303        String convertedLine;
304
305        if (line.startsWith("1/")) {
306            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndDotAndDot(line);
307        } else {
308            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndDotAndNoDot(line);
309        }
310
311        return convertedLine;
312    }
313
314    private static String convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndNoDot(String line) {
315        String convertedLine;
316
317        if (line.startsWith("1/")) {
318            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndNoDotAndDot(line);
319        } else {
320            convertedLine = convertFractionalNumberWithLeadingZeroAndDotAndNoDotAndNoDotAndNoDot(line);
321        }
322
323        return convertedLine;
324    }
325
326    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndDot(String line) {
327        String convertedLine;
328
329        if (line.startsWith("1/")) {
330            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndDotAndDot(line);
331        } else {
332            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndDotAndNoDot(line);
333        }
334
335        return convertedLine;
336    }
337
338    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndNoDot(String line) {
339        String convertedLine;
340
341        if (line.startsWith("1/")) {
342            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndNoDotAndDot(line);
343        } else {
344            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndDotAndNoDotAndNoDot(line);
345        }
346
347        return convertedLine;
348    }
349
350    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndDot(String line) {
351        String convertedLine;
352
353        if (line.startsWith("1/")) {
354            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndDotAndDot(line);
355        } else {
356            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndDotAndNoDot(line);
357        }
358
359        return convertedLine;
360    }
361
362    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDot(String line) {
363        String convertedLine;
364
365        if (line.startsWith("1/")) {
366            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndDot(line);
367        } else {
368            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDot(line);
369        }
370
371        return convertedLine;
372    }
373
374    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndDot(String line) {
375        String convertedLine;
376
377        if (line.startsWith("1/")) {
378            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
379        } else {
380            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
381        }
382
383        return convertedLine;
384    }
385
386    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
387        String convertedLine;
388
389        if (line.startsWith("1/")) {
390            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
391        } else {
392            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
393        }
394
395        return convertedLine;
396    }
397
398    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
399        String convertedLine;
400
401        if (line.startsWith("1/")) {
402            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
403        } else {
404            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
405        }
406
407        return convertedLine;
408    }
409
410    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
411        String convertedLine;
412
413        if (line.startsWith("1/")) {
414            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
415        } else {
416            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
417        }
418
419        return convertedLine;
420    }
421
422    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
423        String convertedLine;
424
425        if (line.startsWith("1/")) {
426            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
427        } else {
428            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
429        }
430
431        return convertedLine;
432    }
433
434    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
435        String convertedLine;
436
437        if (line.startsWith("1/")) {
438            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
439        } else {
440            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
441        }
442
443        return convertedLine;
444    }
445
446    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
447        String convertedLine;
448
449        if (line.startsWith("1/")) {
450            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
451        } else {
452            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
453        }
454
455        return convertedLine;
456    }
457
458    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
459        String convertedLine;
460
461        if (line.startsWith("1/")) {
462            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
463        } else {
464            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
465        }
466
467        return convertedLine;
468    }
469
470    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
471        String convertedLine;
472
473        if (line.startsWith("1/")) {
474            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
475        } else {
476            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
477        }
478
479        return convertedLine;
480    }
481
482    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
483        String convertedLine;
484
485        if (line.startsWith("1/")) {
486            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
487        } else {
488            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
489        }
490
491        return convertedLine;
492    }
493
494    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
495        String convertedLine;
496
497        if (line.startsWith("1/")) {
498            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
499        } else {
500            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
501        }
502
503        return convertedLine;
504    }
505
506    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
507        String convertedLine;
508
509        if (line.startsWith("1/")) {
510            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
511        } else {
512            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
513        }
514
515        return convertedLine;
516    }
517
518    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
519        String convertedLine;
520
521        if (line.startsWith("1/")) {
522            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
523        } else {
524            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
525        }
526
527        return convertedLine;
528    }
529
530    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
531        String convertedLine;
532
533        if (line.startsWith("1/")) {
534            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
535        } else {
536            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
537        }
538
539        return convertedLine;
540    }
541
542    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
543        String convertedLine;
544
545        if (line.startsWith("1/")) {
546            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
547        } else {
548            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
549        }
550
551        return convertedLine;
552    }
553
554    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
555        String convertedLine;
556
557        if (line.startsWith("1/")) {
558            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
559        } else {
560            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
561        }
562
563        return convertedLine;
564    }
565
566    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
567        String convertedLine;
568
569        if (line.startsWith("1/")) {
570            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
571        } else {
572            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
573        }
574
575        return convertedLine;
576    }
577
578    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
579        String convertedLine;
580
581        if (line.startsWith("1/")) {
582            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
583        } else {
584            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
585        }
586
587        return convertedLine;
588    }
589
590    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
591        String convertedLine;
592
593        if (line.startsWith("1/")) {
594            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
595        } else {
596            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
597        }
598
599        return convertedLine;
600    }
601
602    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
603        String convertedLine;
604
605        if (line.startsWith("1/")) {
606            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
607        } else {
608            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
609        }
610
611        return convertedLine;
612    }
613
614    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
615        String convertedLine;
616
617        if (line.startsWith("1/")) {
618            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
619        } else {
620            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
621        }
622
623        return convertedLine;
624    }
625
626    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
627        String convertedLine;
628
629        if (line.startsWith("1/")) {
630            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
631        } else {
632            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
633        }
634
635        return convertedLine;
636    }
637
638    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(String line) {
639        String convertedLine;
640
641        if (line.startsWith("1/")) {
642            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
643        } else {
644            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
645        }
646
647        return convertedLine;
648    }
649
650    private static String convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(String line) {
651        String convertedLine;
652
653        if (line.startsWith("1/")) {
654            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndDot(line);
655        } else {
656            convertedLine = convertFractionalNumberWithLeadingZeroAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDotAndNoDot(line);
657        }
658
659        return convertedLine;
660    }
661
662    private static</b
```

```

55     final String hitNumbers = "([\\d\\.]*[_]*[\\d/+ Measurement.
      NUMBERSIGNS + "]+)";
56     final String hitSpaces = "([\\s°-]+)";
57     final String hitAfter = "(?=\\s\\)\\.,:-]|\$)";
58     StringBuffer units = new StringBuffer();
59     String[] fromUnits = list.getMeasurementTypes();
60     for (int i=0; i<fromUnits.length; i++) {
61         if (i>0) {
62             units.append(" | ");
63         }
64         units.append(Pattern.quote(fromUnits[i]));
65     }
66     String hitUnits = "+units+";
67     switch (type) {
68     case NUMBER_X_NUMBER_UNIT: // 2 x 2 inch
69         return Pattern.compile(hitBefore+hitNumbers+"(\\s*-x
          \\*)\\s*"+hitNumbers+hitSpaces+hitUnits+hitAfter,
          Pattern.CASE_INSENSITIVE);
70     case UNIT_NUMBER: // gas mark 5
71         return Pattern.compile(hitBefore+hitUnits+hitSpaces+
          hitNumbers+hitAfter+"(?!\\s°-)+"+units+"))", Pattern
          .CASE_INSENSITIVE);
72         case NUMBER_UNIT: // 2 cm
73         return Pattern.compile(hitBefore+hitNumbers+hitSpaces+
          hitUnits+hitAfter, Pattern.CASE_INSENSITIVE);
74     }
75     return null;
76   }
77
78   /**
79   * Konverterer mål fra "1 x 1 in" til "2,54 x 2,54 cm"
80   * @param MeasurementList liste over units
81   * @param text opskrift, til konvertering
82   * @return konverteret opskrift
83   */
84   private static String convertNumberXNumber(MeasurementList list,
85     String text) {
86     Pattern p = getMeasurementPattern(list, PatternType.
87       NUMBER_X_NUMBER_UNIT);
88     Matcher m = p.matcher(text);
89     StringBuffer result = new StringBuffer();
90     while (m.find()) {
91       String amount1 = m.group(1);
92       String spacing1 = m.group(2);
93       String amount2 = m.group(3);

```

```

92         String spacing2 = m.group(4);
93         String from      = m.group(5);
94         Measurement unit = list.getMeasurement(from);
95         String newAmount = unit.convert(amount1) + spacing1 +
96                         unit.convert(amount2) + spacing2 + unit.getTo();
97         m.appendReplacement(result, newAmount);
98     }
99     m.appendTail(result);
100    text = result.toString();
101   return text;
102 }
103 /**
104  *
105 * @param MeasurementList liste over units
106 * @param text opskrift, til konvertering
107 * @return konverteret opskrift
108 */
109 private static String convertNumberUnit(MeasurementList list,
110                                         String text) {
111     Pattern p = getMeasurementPattern(list, PatternType.
112                                         NUMBER_UNIT);
113     Matcher m = p.matcher(text);
114     StringBuffer result = new StringBuffer();
115     while (m.find()) {
116         String amount      = m.group(1);
117         String spacing    = m.group(2);
118         String from       = m.group(3);
119         Measurement unit = list.getMeasurement(from);
120         String newAmount = unit.convert(amount) + spacing + unit.
121                           getTo();
122         m.appendReplacement(result, newAmount);
123     }
124 }
125 /**
126 * Konverterer mål fra "gas mark 5" til "10 C"
127 *
128 * @param MeasurementList liste over units
129 * @param text opskrift, til konvertering
130 * @return konverteret opskrift
131 */
132 private static String convertUnitNumber(MeasurementList list,

```

```
133     String text) {
134         Pattern p = getMeasurementPattern(list , PatternType.
135             UNIT_NUMBER);
136         Matcher m = p.matcher(text);
137         StringBuffer result = new StringBuffer();
138         while (m.find()) {
139             String from      = m.group(1);
140             String spacing   = m.group(2);
141             String amount    = m.group(3);
142             Measurement unit = list.getMeasurement(from);
143             String newAmount = unit.convert(amount) + spacing + unit.
144                 getTo();
145             m.appendReplacement(result , newAmount);
146         }
147     }
148
149     /** Konvertér mål i en opskrift fra en type til en anden
150      *
151      * @param text opskrift , til konvertering
152      * @return konverteret opskrift
153      */
154     public static String convert(String text) {
155         MeasurementList list = new MeasurementList();
156         text = convertNumberXNumber(list , text);
157         text = convertUnitNumber(list , text);
158         text = convertNumberUnit(list , text);
159         return text;
160     }
161 }
162 }
```

E.7 FileIOTest.java

```
1
2
3 import java.io.File;
4
5 import junit.framework.TestCase;
6
7 public class FileIOTest extends TestCase {
8
9 }
```

```

10     /**
11      * Bemærk at testen benytter Windows-linjeskift og fungerer
12      * derfor kun i Windows
13     */
14     public final void testReadFile() {
15         File testfile = new File("texts/testtext.txt");
16         String inputFromReader = FileIO.readFile(testfile);
17         assertEquals("Test\r\n12345\r\nabcde" , inputFromReader);
18     }
19
20     /**
21      * Bemærk at denne test fejler , hvis ikke FileIO.readFile
22      * metoden er i orden , de to tests er derfor forbundne
23      *
24      * Bemærk at testen benytter Windows-linjeskift og fungerer
25      * derfor kun i Windows
26      */
27     public final void testWriteFile() {
28         File testfile2 = new File("texts/testwrite.txt");
29         FileIO.writeFile(testfile2 , "Jeg_skal_skrives_til_en_testfil");
30         String inputFromReader2 = FileIO.readFile(testfile2);
31         assertEquals("Jeg_skal_skrives_til_en_testfil" ,
32                     inputFromReader2);
33
34         File testfile3 = new File("texts/testwrite2.txt");
35         FileIO.writeFile(testfile3 , "Jeg_skal_skrives\r\n til_en_"
36                         + "testfil");
37         String inputFromReader3 = FileIO.readFile(testfile3);
38         assertEquals("Jeg_skal_skrives\r\n til_en_testfil" ,
39                     inputFromReader3);
40     }

```

E.8 MeasurementTest.java

```

1 import junit.framework.TestCase;
2
3 public class MeasurementTest extends TestCase {
4
5     private Measurement inchToCm;

```

```

6     private Measurement poundToGram;
7
8
9     protected void setUp()
10    {
11         inchToCm = new Measurement("inch" , "cm" , (double)2.54 , 0);
12         poundToGram = new Measurement("lb\tgram\t454\t0");
13    }
14
15    public final void testGetFrom() {
16        assertTrue(!inchToCm.equals(null));
17        assertEquals("inch" , inchToCm.getFrom());
18        assertTrue(!poundToGram.equals(null));
19        assertEquals("lb" , poundToGram.getFrom());
20    }
21
22    public final void testConvert() {
23        assertTrue(!inchToCm.equals(null));
24        assertEquals("2,54" , inchToCm.convert("1.0"));
25        assertEquals("5,08" , inchToCm.convert("2.0"));
26        assertEquals("3,81" , inchToCm.convert("1½"));
27        assertEquals("6,98" , inchToCm.convert("2¾"));
28        assertEquals("2,54" , inchToCm.convert("1.0"));
29        assertEquals("5,72" , inchToCm.convert("2¼"));
30
31        assertTrue(!poundToGram.equals(null));
32        assertEquals("454" , poundToGram.convert("1.0"));
33        assertEquals("908" , poundToGram.convert("2.0"));
34        assertEquals("681" , poundToGram.convert("1½"));
35        assertEquals("1.248,5" , poundToGram.convert("2¾"));
36        assertEquals("454" , poundToGram.convert("1.0"));
37        assertEquals("454" , poundToGram.convert("1"));
38    }
39
40    /**
41     * Dette er en vigtig test , for programmets funktionalitet .
42     *
43     */
44    public final void testConvertToNumber() {
45        assertTrue(!inchToCm.equals(null));
46
47        //Først alle special karaktererne med og uden et kombineret
48        //tal , men uden mellemrum
48        assertEquals((double)(5/2.) , Measurement.convertToNumber("2½"));

```

```

49     assertEquals((double)(23/4.) , Measurement.convertToNumber("5
      3
      4"));
50     assertEquals((double)(17/4.) , Measurement.convertToNumber("4
      1
      4"));
51     assertEquals((double)(1/2.) , Measurement.convertToNumber("1
      2
      "));
52     assertEquals((double)(3/4.) , Measurement.convertToNumber("3
      4
      "));
53     assertEquals((double)(1/4.) , Measurement.convertToNumber("1
      4
      "));
54     assertEquals((double)(7/3.) , Measurement.convertToNumber("2\
      u2153")); // 2 1/3
55     assertEquals((double)(11/3.) , Measurement.convertToNumber("
      3\u2154")); // 3 2/3
56     assertEquals((double)(9/8.) , Measurement.convertToNumber("1\
      u215B")); // 1 1/8
57     assertEquals((double)(19/8.) , Measurement.convertToNumber("
      2\u215C")); // 2 3/8
58     assertEquals((double)(29/8.) , Measurement.convertToNumber("
      3\u215D")); // 3 5/8
59     assertEquals((double)(39/8.) , Measurement.convertToNumber("
      4\u215E")); // 4 7/8
60     assertEquals((double)(1/3.) , Measurement.convertToNumber("\
      u2153")); // 1/3
61     assertEquals((double)(2/3.) , Measurement.convertToNumber("\
      u2154")); // 2/3
62     assertEquals((double)(1/8.) , Measurement.convertToNumber("\
      u215B")); // 1/8
63     assertEquals((double)(3/8.) , Measurement.convertToNumber("\
      u215C")); // 3/8
64     assertEquals((double)(5/8.) , Measurement.convertToNumber("\
      u215D")); // 5/8
65     assertEquals((double)(7/8.) , Measurement.convertToNumber("\
      u215E")); // 7/8
66
67     //Så tager vi dem med et mellermrum
68     assertEquals((double)(7/3.) , Measurement.convertToNumber("2 \
      \u2153")); // 2 1/3
69     assertEquals((double)(11/3.) , Measurement.convertToNumber("3 \
      \u2154")); // 3 2/3
70     assertEquals((double)(9/8.) , Measurement.convertToNumber("1 \
      \u215B")); // 1 1/8
71     assertEquals((double)(19/8.) , Measurement.convertToNumber("2 \
      \u215C")); // 2 3/8
72     assertEquals((double)(29/8.) , Measurement.convertToNumber("3
      1
      8
      "));

```

```

    "½\u215D")); // 3 5/8
73     assertEquals((double)(39/8.) , Measurement.convertToNumber("4
    "½\u215E")); // 4 7/8
74     assertEquals((double)(5/2.) , Measurement.convertToNumber("2
    ½"));
75     assertEquals((double)(23/4.) , Measurement.convertToNumber("5
    ¾"));
76     assertEquals((double)(17/4.) , Measurement.convertToNumber("4
    ¼"));
77
78 }
79
80
81 public final void testGetTo() {
82     assertTrue(!inchToCm.equals(null));
83     assertEquals("cm" , inchToCm.getTo());
84     assertTrue(!poundToGram.equals(null));
85     assertEquals("gram" , poundToGram.getTo());
86 }
87
88 /*
89  * Class under test for String toString()
90  */
91 public final void testToString() {
92     assertTrue(!inchToCm.equals(null));
93     assertEquals("inch\t\tcm\t\t2.54\t\t0" , inchToCm.toString())
94         ;
95     assertTrue(!poundToGram.equals(null));
96     assertEquals("lb\t\tgram\t\t454.0\t\t0" , poundToGram.
97         toString());
98 }
```

E.9 RecipeConverterTest.java

```

1 import junit.framework.TestCase;
2
3 public class RecipeConverterTest extends TestCase {
4
5     public final void testpunktum()
6     {
7         //Test for forkortelser med og uden punktum
8         assertEquals(RecipeConverter.convert("Test_10_in_skal_
    omregnes") , "Test_25,4_cm_skal_omregnes");
9         assertEquals(RecipeConverter.convert("Test_10_in._skal_
```

```

10         omregnes") , "Test_25,4_cm._skal_omregnes");
11     assertEquals(RecipeConverter.convert("Test_10_inches_skal_
12         omregnes") , "Test_25,4_centimeter_skal_omregnes");
13     assertEquals(RecipeConverter.convert("Test_10_inch._skal_
14         omregnes") , "Test_25,4_centimeter._skal_omregnes");
15 }
16
17     public final void testStoreBogstaver()
18 {
19     // Test for store og små bogstaver
20     assertEquals(RecipeConverter.convert("Test_10_In_skal_
21         omregnes") , "Test_25,4_cm_skal_omregnes");
22     assertEquals(RecipeConverter.convert("Test_10_In._skal_
23         omregnes") , "Test_25,4_cm._skal_omregnes");
24     assertEquals(RecipeConverter.convert("Test_10_Inches_skal_
25         omregnes") , "Test_25,4_centimeter_skal_omregnes");
26     assertEquals(RecipeConverter.convert("Test_10_Inch._skal_
27         omregnes") , "Test_25,4_centimeter._skal_omregnes");
28 }
29
30     public final void testWhitespace()
31 {
32     //Test for ekstra mellemrum tab etc.
33     assertEquals(RecipeConverter.convert("Test_10_in_skal_
34         omregnes") , "Test_25,4_cm_skal_omregnes");
35     assertEquals(RecipeConverter.convert("Test_10_in_skal_
36         omregnes") , "Test_25,4_cm_skal_omregnes");
37     assertEquals(RecipeConverter.convert("Test\n10_in_skal_
38         omregnes") , "Test\n25,4_cm_skal_omregnes");
39     assertEquals(RecipeConverter.convert("Test\n10_in_skal_
40         omregnes") , "Test\n25,4_cm_skal_omregnes");
41     assertEquals(RecipeConverter.convert("Test\t10_in_skal_
42         omregnes") , "Test\t25,4_n_cm_skal_omregnes");
43     assertEquals(RecipeConverter.convert("Test\t10_tin_skal_
44         omregnes") , "Test\t25,4_tcm_skal_omregnes");
45 }
46
47     public final void testparanteser()
48 {
49     //Test for brøker og sammensatte tal
50     assertEquals(RecipeConverter.convert("test_7_længder_(1/2_in)
51         _skal_omregnes") , "test_7_længder_(1,27_cm)_skal_omregnes
52         ");
53     assertEquals(RecipeConverter.convert("test_(1½_in.)_skal_
54         omregnes") , "test_(3,81_cm.)_skal_omregnes");
55 }
```

```

39         assertEquals(RecipeConverter.convert("1 1/2 inches")_skal_
40                     "omregnes") , "3,81 centimeter"_skal_omregnes");
41         assertEquals(RecipeConverter.convert("test_(4 1/4 inch)_skal_
42                     "omregnes") , "test_(10,8 centimeter)_skal_omregnes");
43     }
44
45
46     public final void testkomma()
47     {
48         //Test decimaltal
49         assertEquals(RecipeConverter.convert("Test_1.5_in_skal_
50                     "omregnes") , "Test_3,81_cm_skal_omregnes");
51         assertEquals(RecipeConverter.convert("Test_1.5_in._skal_
52                     "omregnes") , "Test_3,81_cm._skal_omregnes");
53         assertEquals(RecipeConverter.convert("Test_1.5_inches_skal_
54                     "omregnes") , "Test_3,81_centimeter_skal_omregnes");
55         assertEquals(RecipeConverter.convert("Test_1.5_inch._skal_
56                     "omregnes") , "Test_3,81_centimeter._skal_omregnes");
57     }
58
59     public final void testTalFørst()
60     {
61         //Test hvis tallet står aller først i strengen
62         assertEquals(RecipeConverter.convert("1.5_in_skal_omregnes")
63                     , "3,81_cm_skal_omregnes");
64         assertEquals(RecipeConverter.convert("1.5_in._skal_omregnes")
65                     , "3,81_cm._skal_omregnes");
66         assertEquals(RecipeConverter.convert("1.5_inches_skal_
67                     "omregnes") , "3,81_centimeter_skal_omregnes");
68         assertEquals(RecipeConverter.convert("1.5_inch._skal_omregnes"
69                     ) , "3,81_centimeter._skal_omregnes");
70     }
71
72     public final void testBrøktal()
73     {
74         //Test for brøker og sammensatte tal
75         assertEquals(RecipeConverter.convert("test_1 1/2_in_skal_omregnes"
76                     ) , "test_1,27_cm_skal_omregnes");
77         assertEquals(RecipeConverter.convert("test_1 1/2_in._skal_
78                     "omregnes") , "test_3,81_cm._skal_omregnes");
79         assertEquals(RecipeConverter.convert("1 1/2_inches_skal_
80                     "omregnes") , "3,81_centimeter_skal_omregnes");
81         assertEquals(RecipeConverter.convert("test_4 1/4_inch._skal_
82                     "omregnes") , "test_10,8_cm._skal_omregnes");
83     }

```

```

    omregnes") , "test_10,8_centimeter._skal_omregnes");
71    assertEquals(RecipeConverter.convert("23/4inches_skal_omregnes
      ") , "6,98_centimeter_skal_omregnes");
72    assertEquals(RecipeConverter.convert("11/2inches_skal_
      omregnes") , "3,81_centimeter_skal_omregnes");
73    assertEquals(RecipeConverter.convert("test_17/4inch._skal_
      omregnes") , "test_10,8_centimeter._skal_omregnes");
74    assertEquals(RecipeConverter.convert("11/4inches_skal_
      omregnes") , "6,98_centimeter_skal_omregnes");
75  }
76
77  public final void testDobbelteOmregninger()
78  {
79    //Test for dobbelt tal, dvs fx. 5 * 3 in. tester også med og
     uden mellemrum
80    assertEquals(RecipeConverter.convert("test_2_*_3_in_skal_
      omregnes") , "test_5,08_*_7,62_cm_skal_omregnes");
81    assertEquals(RecipeConverter.convert("test_2*3_in._skal_
      omregnes") , "test_5,08*7,62_cm._skal_omregnes");
82    assertEquals(RecipeConverter.convert("11/2*_3_inches_skal_
      omregnes") , "3,81*_7,62_centimeter_skal_omregnes");
83    assertEquals(RecipeConverter.convert("23/4*_41/4inch._skal_
      omregnes") , "6,98*_10,8_centimeter._skal_omregnes");
84  }
85
86  public final void testFraOgTilOmregninger()
87  {
88    //Test for dobbelt tal, dvs fx. 5 * 3 in. tester også med og
     uden mellemrum
89    assertEquals(RecipeConverter.convert("test_2_-_3_in_skal_
      omregnes") , "test_5,08_-_7,62_cm_skal_omregnes");
90    assertEquals(RecipeConverter.convert("test_2-3_in._skal_
      omregnes") , "test_5,08-7,62_cm._skal_omregnes");
91    assertEquals(RecipeConverter.convert("11/2-_3_inches_skal_
      omregnes") , "3,81_-_7,62_centimeter_skal_omregnes");
92    assertEquals(RecipeConverter.convert("23/4-_41/4inch._skal_
      omregnes") , "6,98_-_10,8_centimeter._skal_omregnes");
93  }
94
95
96
97  public final void testEnhedFørst()
98  {
99    //Test for muligheden for at enheden står først
100   assertEquals(RecipeConverter.convert("test_in_3_skal_omregnes

```

```

101         ") , "test_7,62_cm_skal_omregnes");
102         assertEquals(RecipeConverter.convert("test_in_3½_skal_
103             omregnes") , "test_8,89_cm_skal_omregnes");
104         assertEquals(RecipeConverter.convert("inch_2_skal_omregnes")
105             , "5,08_centimeter_skal_omregnes");
106     }
107     public final void testKorteStrenge()
108     {
109         //Test for helt korte strenge
110         assertEquals(RecipeConverter.convert("3_in") , "7,62_cm");
111         assertEquals(RecipeConverter.convert("3½_in") , "8,89_cm");
112         assertEquals(RecipeConverter.convert("3½_in.") , "8,89_cm.
113             ");
114         assertEquals(RecipeConverter.convert("2_inches") , "5,08_
115             centimeter");
116     }
117     public final void testNegative()
118     {
119         //Disse tilfælde skulle ikke gerne konverteres
120         assertEquals(RecipeConverter.convert("Test3_in") , "Test3_in"
121             );
122         assertEquals(RecipeConverter.convert("42") , "42");
123         assertEquals(RecipeConverter.convert("3½in") , "3½in");
124         assertEquals(RecipeConverter.convert("3a½in.") , "3a½in.
125             ");
126         assertEquals(RecipeConverter.convert("(2)_inches") , "(2)_inches");
127     }
128 }
```

E.10 Case1Test.java

```

1 import junit.framework.TestCase;
2
3 public class Case1Test extends TestCase {
4
5
6
7     /**
8      * Denne test er for at undersøge om konverteringen af
9      * 2 x 3 eller 3 * 4 tilfældene omregnes korrekt
10     */
11 }
```

```
12  public final void testCase1()
13  {
14
15      //Ingen fejl
16      assertEquals(RecipeConverter.convert("Test_10_x_5_inches."),
17                  "Test_25,4_x_12,7_centimeter.");
18
19      //fejl 5
20      assertEquals(RecipeConverter.convert("Test_10_x_5_abcinches."),
21                  "Test_10_x_5abcinches.");
22
23      //fejl 7
24      assertEquals(RecipeConverter.convert("Test_10_x_5_inchesnej"),
25                  "Test_10_x_5_inchesnej");
26
27      // fejl 5 , 6
28      assertEquals(RecipeConverter.convert("Test_10_x_5Abcikke."),
29                  "Test_10_x_5Abcikke.");
30
31      // fejl 4 , 6
32      assertEquals(RecipeConverter.convert("Test_10_x_ejHeller_ikke."),
33                  "Test_10_x_ejHeller_ikke.");
34
35      // fejl 4 , 5 , 6 , 7
36      assertEquals(RecipeConverter.convert("Test_10_x_ejHellerAbcikkenej"),
37                  "Test_10_x_ejHellerAbcikkenej");
38
39      // fejl 4 , 7
40      assertEquals(RecipeConverter.convert("Test_10_x_ejHeller_inchesnej"),
41                  "Test_10_x_ejHeller_inchesnej");
42
43      // fejl 4 , 5 , 7
44      assertEquals(RecipeConverter.convert("Test_10_x_ejHellerAbcincchesnej"),
45                  "Test_10_x_ejHellerAbcincchesnej");
46
47      // fejl 3 , 7
48      assertEquals(RecipeConverter.convert("Test_10DurIkke5_inchesnej"),
49                  "Test_10DurIkke5_inchesnej");
50
51      // fejl 3 , 5 , 6 , 7
52      assertEquals(RecipeConverter.convert("Test_10_DurIkke5Abcikkenej"),
53                  "Test_10DurIkke5Abcikkenej");
54
55      // fejl 3 , 6
```

```
46     assertEquals(RecipeConverter.convert("Test_10DurIkke5_ikke_")
47                 , "Test_10DurIkke5_ikke_");
48     // fejl 3 , 5 , 6
49     assertEquals(RecipeConverter.convert("Test_10Durikke5Abcikke_")
50                 , "Test_10Durikke5Abcikke_");
51     // fejl 3 , 4
52     assertEquals(RecipeConverter.convert("Test_10DurikkejHeller_
53             inches_")
54                 , "Test_10DurikkejHeller_inches_");
55     // fejl 3 , 4 , 5
56     assertEquals(RecipeConverter.convert("Test_10
57             DurIkkejHellerAbcincches_")
58                 , "Test_10
59             DurIkkejHellerAbcincches_");
60     // fejl 3 , 4 , 5 , 6 , 7
61     assertEquals(RecipeConverter.convert("Test_10
62             DurIkkejHellerAbcikkenej")
63                 , "Test_10
64             DurIkkejHellerAbcikkenej");
65     // fejl 2 , 6 , 7
66     assertEquals(RecipeConverter.convert("Test_durIkke_x_5_
67             ikkenej")
68                 , "Test_durIkke_x_5_ikkenej");
69     // fejl 2 , 5 , 6 ,7
70     assertEquals(RecipeConverter.convert("Test_durIkke_x_5_
71             Abcikkenej")
72                 , "Test_durIkke_x_5Abcikkenej");
73     // fejl 2 , 7
74     assertEquals(RecipeConverter.convert("Test_durIkke_x_5_
75             inchesnej")
76                 , "Test_durIkke_x_5inchesnej");
77     // fejl 2 , 5 , 7
78     assertEquals(RecipeConverter.convert("Test_durIkke_x_5_
79             Abcinchesnej")
80                 , "Test_durIkke_x_5Abcinchesnej");
81     // fejl 2 , 4
82     assertEquals(RecipeConverter.convert("Test_durIkke_x_ejHeller_
83             _inches_")
84                 , "Test_durIkke_x_ejHeller_inches_");
```

```

78      // fejl 2 , 4 , 5 , 6
79      assertEquals(RecipeConverter.convert("Test_durIkke_x_
        ejHellerAbcikke."), "Test_durIkke_x_ejHellerAbcikke.");
80
81      // fejl 2 , 4 , 6
82      assertEquals(RecipeConverter.convert("Test_durIkke_x_ejHeller
        _ikkke."), "Test_durIkke_x_ejHeller_ikkke.");
83
84      // fejl 2 , 4, 5 , 6, 7
85      assertEquals(RecipeConverter.convert("Test_durIkke_x_
        ejHellerAbcikkenej"), "Test_durIkke_x_ejHellerAbcikkenej"
        );
86
87      // fejl 2 , 3
88      assertEquals(RecipeConverter.convert("Test_durIkkedurikke5_
        inches."), "Test_durIkkedurikke5_inches.");
89
90      // fejl 2 , 3 , 5, 7
91      assertEquals(RecipeConverter.convert("Test_d
        urIkkedurikke5Abcinchesnej"), "Test_d
        urIkkedurikke5Abcinchesnej");
92
93      // fejl 2 , 3 , 7
94      assertEquals(RecipeConverter.convert("Test_durIkkedurikke5_
        inchesnej"), "Test_durIkkedurikke5_inchesnej");
95
96      // fejl 2 , 3 , 5 , 6 , 7
97      assertEquals(RecipeConverter.convert("Test_d
        urIkkedurikke5Abcikkenej"), "Test_d
        urIkkedurikke5Abcikkenej");
98
99      // fejl 2 , 3 , 4 , 6 , 7
100     assertEquals(RecipeConverter.convert("Test_d
        urIkkedurikkejHeller_ikkenej"), "Test_d
        urIkkedurikkejHeller_ikkenej");
101
102     // fejl 2 , 3 , 4 , 5 , 6 , 7
103     assertEquals(RecipeConverter.convert("Test_d
        urIkkedurikkejHellerAbcikkenej"), "Test_d
        urIkkedurikkejHellerAbcikkenej");
104
105     // fejl 2 , 3 , 4
106     assertEquals(RecipeConverter.convert("Test_d
        urIkkeDurIkkejHeller_inches."), "Test_d
        urIkkeDurIkkejHeller_inches.");

```

```
107  
108     // fejl 2 , 3 , 4 , 5  
109     assertEquals(RecipeConverter.convert("Test_  
110         durIkkeDurIkkeejHellerAbcinches .") , "Test_  
111         durIkkeDurIkkeejHellerAbcinches .");  
112  
113     // fejl 1  
114     assertEquals(RecipeConverter.convert("Test10_x_5_inches .") ,  
115                     "Test10_x_12,7_centimeter .");  
116  
117     // fejl 1 , 5 , 6  
118     assertEquals(RecipeConverter.convert("Test10_x_5Abcikke .") ,  
119                     "Test10_x_5Abcikke .");  
120  
121     // fejl 1 , 5 , 6 , 7  
122     assertEquals(RecipeConverter.convert("Test10_x_5Abcikkenej") ,  
123                     "Test10_x_5Abcikkenej");  
124  
125     // fejl 1 , 4 , 7  
126     assertEquals(RecipeConverter.convert("Test10_x_ejHeller_  
127             inchesnej") , "Test10_x_ejHeller_inchesnej");  
128  
129     // fejl 1 , 4 , 5 , 7  
130     assertEquals(RecipeConverter.convert("Test10_x_ejHeller_  
131             ejHellerAbcinchesnej") , "Test10_x_ejHellerAbcinchesnej");  
132  
133     // fejl 1 , 4 , 5 , 6 , 7  
134     assertEquals(RecipeConverter.convert("Test10_x_ejHeller_  
135             Abcikkenej") , "Test10_x_ejHellerAbcikkenej");  
136  
137     // fejl 1 , 3 , 6  
138     assertEquals(RecipeConverter.convert("Test10DurIkke5_ikke .") ,  
139                     "Test10DurIkke5_ikke .");  
140  
141     // fejl 1 , 3 , 5 , 6  
142     assertEquals(RecipeConverter.convert("Test10DurIkke5Abcikke .") ,  
143                     "Test10DurIkke5Abcikke .");
```

```

140
141     // fejl 1 , 3
142     assertEquals(RecipeConverter.convert("Test10DurIkke5_inches ."
143                           " ) , "Test10DurIkke5_inches .");
144
145     // fejl 1 , 3 , 5
146     assertEquals(RecipeConverter.convert("Test10DurIkke5Abcinches"
147                           ".") , "Test10DurIkke5Abcinches .");
148
149     // fejl 1 , 3 , 4
150     assertEquals(RecipeConverter.convert("Test10DurIkkeejHeller_"
151                           " inches .") , "Test10DurIkkeejHeller_inches .");
152
153     // fejl 1 , 3 , 4 , 5 , 6 , 7
154     assertEquals(RecipeConverter.convert("Test10DurIkkeejHellerAbcikkenej")
155                           , "Test10DurIkkeejHellerAbcikkenej");
156
157     // fejl 1 , 3 , 4 , 5 , 6
158     assertEquals(RecipeConverter.convert("Test10DurIkkeejHellerAbcikke .")
159                           , "Test10DurIkkeejHellerAbcikke .");
160
161     // fejl 1 , 2 , 7
162     assertEquals(RecipeConverter.convert("TestdurIkke_x_5_"
163                           " inchesnej") , "TestdurIkke_x_5_inchesnej");
164
165     // fejl 1 , 2 , 5 , 7
166     assertEquals(RecipeConverter.convert("TestdurIkke_x_5_"
167                           " Abcinchesnej") , "TestdurIkke_x_5Abcinchesnej");
168
169     // fejl 1 , 2 , 5 , 6
170     assertEquals(RecipeConverter.convert("TestDurIkke_x_5_inches ."
171                           " ) , "TestDurIkke_x_12,7_centimeter .");
172
173     // fejl 1 , 2 , 5 , 6
174     assertEquals(RecipeConverter.convert("TestdurIkke_x_5Abcikke .")
175                           , "TestdurIkke_x_5Abcikke .");
176
177     // fejl 1 , 2 , 4 , 6
178     assertEquals(RecipeConverter.convert("TestdurIkke_x_ejHeller_"
179                           " inchesnej") , "TestdurIkke_x_ejHeller_inches .");

```

```
    ikke .") , "TestdurIkke_x_ejHeller_ikke .");
173
174     // fejl 1 , 2 , 4 , 5 , 6
175     assertEquals(RecipeConverter.convert("TestdurIkke_x_
176         ejHellerAbcikke .") , "TestdurIkke_x_ejHellerAbcikke .");
177
178     // fejl 1 , 2 , 4
179     assertEquals(RecipeConverter.convert("TestdurIkke_x_ejHeller_
180         inches .") , "TestdurIkke_x_ejHeller_inches .");
181
182     // fejl 1 , 2 , 4 , 5 , 7
183     assertEquals(RecipeConverter.convert("TestdurIkke_x_ejHellerAbcincchesnej") , "TestdurIkke_x_ejHellerAbcincchesnej");
184
185     // fejl 1 , 2 , 6 , 7
186     assertEquals(RecipeConverter.convert("TestdurIkke_x_5_ikkenej")
187         , "TestdurIkke_x_5_ikkenej");
188
189     // fejl 1 , 2 , 3 , 5 , 6 , 7
190     assertEquals(RecipeConverter.convert("TestdurIkkeDurIkke5Abcikkenej") , "TestdurIkkeDurIkke5Abcikkenej");
191
192     // fejl 1 , 2 , 3 , 4
193     assertEquals(RecipeConverter.convert("TestdurIkkeDurIkkeejHellerAbcikkenej") , "TestdurIkkeDurIkkeejHellerAbcikkenej");
194
195     // fejl 1 , 2 , 3 , 4 , 5
196     assertEquals(RecipeConverter.convert("TestdurIkkeDurIkkeejHellerAbcincches .") , "TestdurIkkeDurIkkeejHellerAbcincches .");
197
198     // fejl 1 , 3 , 4 , 6 , 7
199     assertEquals(RecipeConverter.convert("Test10DurIkkeejHeller_ikkenej") , "Test10DurIkkeejHeller_ikcenej");
200
201     // fejl 1 , 2 , 3 , 4 , 7
202     assertEquals(RecipeConverter.convert("
```

```
203     TestdurIkkeDurIkkeejHeller_inchesnej") , "  
204     TestdurIkkeDurIkkeejHeller_inchesnej");  
205     // fejl 1 , 2 , 3 , 4 , 5 , 6  
206     assertEquals(RecipeConverter.convert("TestdurIkkeDurIkkeejHellerAbcikke_") , "  
207     TestdurIkkeDurIkkeejHellerAbcikke_");  
208 }  
209  
210 }
```

E.11 Case2Test.java

```

1 import junit.framework.TestCase;
2
3 public class Case2Test extends TestCase {
4
5     /**
6      * Metode for test af fejl på alle grænser når en unit skal
7      * konverteres .
8      * Tester om kravene hitBefore+hitUnits+hitSpaces+hitNumbers
9      * +hitAfter+"(?![\s°-]+("+units+"))" blir opfyldt
10     * "(?![\s°-]+("+units+"))" betyr at der ikke må stå et
11     * mellemrom efterfulgt
12     * af en enhed vi kender
13     * ue = ukendt enhed
14     */
15
16     public final void testUnit_Number() {
17
18         //Ingen fejl på hitBefore+hitUnits+hitSpaces+
19         //hitNumbers+hitAfter+"(?![\s°-]+("+units+"))"
20         assertEquals(RecipeConverter.convert("Test_Gas_mark_5_
21             _ue_skal_omregnes") , "Test_191,25_°Celsius_ue_
22             skal_omregnes");
23
24         //Fejl på hitBefore
25         assertEquals(RecipeConverter.convert("TestGas_mark_5_
26             ue_skal_ikke_omregnes") , "TestGas_mark_5_ue_skal_
27             ikke_omregnes");
28
29         //Fejl på hitUnits
30         assertEquals(RecipeConverter.convert("Test_Gassmark_5_

```

```

        „ue_skal_ikke_omregnes") , "Test_Gassmark_5_ue_
        skal_ikke_omregnes");

24
25    // Fejl på hitSpaces
26    assertEquals(RecipeConverter.convert("Test_Gas_mark5_
        ue_skal_ikke_omregnes") , "Test_Gas_mark5_ue_skal_
        ikke_omregnes");

27
28    // Fejl på hitNumbers
29    assertEquals(RecipeConverter.convert("Test_Gas_mark_a
        _ue_skal_ikke_omregnes") , "Test_Gas_mark_a_ue_
        skal_ikke_omregnes");

30
31    // Fejl på hitAfter
32    assertEquals(RecipeConverter.convert("Test_Gas_mark_5
        ue_skal_ikke_omregnes") , "Test_Gas_mark_5ue_skal_
        ikke_omregnes");

33
34    // Fejl på "(?![\s°-]+(+units+))"
35    assertEquals(RecipeConverter.convert("Test_Gas_mark_5
        _in_der_in_skal_omregnes") , "Test_Gas_mark_12,7_
        cm_der_in_skal_omregnes");

36
37    // Fejl på hitBefore+hitUnits
38    assertEquals(RecipeConverter.convert("TestGassmark_5_
        ue_skal_ikke_omregnes") , "TestGassmark_5_ue_skal_
        ikke_omregnes");

39
40    // Fejl på hitBefore+hitSpaces
41    assertEquals(RecipeConverter.convert("TestGas_mark5_
        ue_skal_ikke_omregnes") , "TestGas_mark5_ue_skal_
        ikke_omregnes");

42
43    // Fejl på hitBefore+hitNumbers
44    assertEquals(RecipeConverter.convert("TestGas_mark_a_
        ue_skal_ikke_omregnes") , "TestGas_mark_a_ue_skal_
        ikke_omregnes");

45
46    // Fejl på hitBefore+hitAfter
47    assertEquals(RecipeConverter.convert("TestGas_mark_5
        ue_skal_ikke_omregnes") , "TestGas_mark_5ue_skal_
        ikke_omregnes");

48
49    // Fejl på hitBefore +"(?![\s°-]+(+units+))"
50    assertEquals(RecipeConverter.convert("TestGas_mark_5_

```

```

51     in_der_in_skal_omregnes") , "TestGas_mark_12,7_cm_
52     der_in_skal_omregnes");
53
54     // Fejl på hitUnits+hitSpaces
55     assertEquals(RecipeConverter.convert("Test_Gassmark5_
56         ue_skal_ikke_omregnes") , "Test_Gassmark5_ue_skal_
57         ikke_omregnes");
58
59     // Fejl på hitUnits+hitNumbers
60     assertEquals(RecipeConverter.convert("Test_Gassmark_a_
61         ue_skal_ikke_omregnes") , "Test_Gassmark_a_ue_
62         skal_ikke_omregnes");
63
64     // Fejl på hitUnits+"(?![\\s°-]+("+units+"))"
65     assertEquals(RecipeConverter.convert("Test_Gassmark_5_
66         _in_der_in_skal_omregnes") , "Test_Gassmark_12,7_
67         cm_der_in_skal_omregnes");
68
69     // Fejl på hitSpaces+hitNumbers
70     assertEquals(RecipeConverter.convert("Test_Gas_marka_
71         ue_skal_ikke_omregnes") , "Test_Gas_marka_ue_skal_
72         ikke_omregnes");
73
74     // Fejl på hitSpaces+"(?![\\s°-]+("+units+"))"
75     assertEquals(RecipeConverter.convert("Test_Gas_mark5_
76         in_skal_ikke_omregnes") , "Test_Gas_mark5_in_skal_
77         ikke_omregnes");
78
79     // Fejl på hitNumbers+hitAfter
80     assertEquals(RecipeConverter.convert("Test_Gas_mark_
81         aue_skal_ikke_omregnes") , "Test_Gas_mark_aue_skal_
82         ikke_omregnes");
83
84     // Fejl på hitNumbers+"(?![\\s°-]+("+units+"))"
85     assertEquals(RecipeConverter.convert("Test_Gas_mark_a_
86         ue_skal_ikke_omregnes") , "Test_Gas_mark_a_ue_skal_
87         ikke_omregnes");
88
89     // Fejl på hitNumbers+"(?![\\s°-]+("+units+"))"
90     assertEquals(RecipeConverter.convert("Test_Gas_mark_a_
91         ue_skal_ikke_omregnes") , "Test_Gas_mark_a_ue_skal_
92         ikke_omregnes");
93
94     // Fejl på hitNumbers+"(?![\\s°-]+("+units+"))"
95     assertEquals(RecipeConverter.convert("Test_Gas_mark_a_
96         ue_skal_ikke_omregnes") , "Test_Gas_mark_a_ue_skal_
97         ikke_omregnes");

```

```
    "in_skal_ikke_omregnes") , "Test_Gas_mark_a_in_
    skal_ikke_omregnes");
78
79    // Fejl på hitAfter + "(?![\s° -]+(" + units + "))"
80    assertEquals(RecipeConverter.convert("Test_Gas_mark_5
        in_skal_ikke_omregnes") , "Test_Gas_mark_5in_skal_
        ikke_omregnes");
81
82    // Fejl på hitBefore+hitUnits+hitSpaces
83    assertEquals(RecipeConverter.convert("TestGassmark5_
        ue_skal_ikke_omregnes") , "TestGassmark5_ue_skal_
        ikke_omregnes");
84
85    // Fejl på hitBefore+hitUnits+hitNumbers
86    assertEquals(RecipeConverter.convert("TestGassmark_a_
        ue_skal_ikke_omregnes") , "TestGassmark_a_ue_skal_
        ikke_omregnes");
87
88    // Fejl på hitBefore+hitUnits+hitAfter
89    assertEquals(RecipeConverter.convert("TestGassmark_5
        ue_skal_ikke_omregnes") , "TestGassmark_5ue_skal_
        ikke_omregnes");
90
91    // Fejl på hitBefore+hitUnits +"(?![\s° -]+(" + units + "))
92    assertEquals(RecipeConverter.convert("TestGassmark_5_
        in_der_in_skal_omregnes") , "TestGassmark_12,7_cm_
        der_in_skal_omregnes");
93
94    // Fejl på hitBefore+hitSpaces+hitNumbers
95    assertEquals(RecipeConverter.convert("TestGas_marka_
        ue_skal_ikke_omregnes") , "TestGas_marka_ue_skal_
        ikke_omregnes");
96
97    // Fejl på hitBefore+hitSpaces+hitAfter
98    assertEquals(RecipeConverter.convert("TestGas_mark5ue_
        skal_ikke_omregnes") , "TestGas_mark5ue_skal_ikke_
        omregnes");
99
100   // Fejl på hitBefore+hitSpaces +"(?![\s° -]+(" + units + "))
101  assertEquals(RecipeConverter.convert("TestGas_mark5_
        in_skal_ikke_omregnes") , "TestGas_mark5in_skal_
        ikke_omregnes");
102
```

```

103      // Fejl på hitBefore+hitNumbers+hitAfter +"(?![\\s°
104      // -]+("+units+"))"
104      assertEquals(RecipeConverter.convert("TestGas_mark_
105          ain_skal_ikke_omregnes") , "TestGas_mark_ain_skal_
106          ikke_omregnes");
107      // Fejl på hitBefore+hitNumbers +"(?![\\s° -]+("+units
108          +"))"
107      assertEquals(RecipeConverter.convert("TestGas_mark_a_
109          in_skal_ikke_omregnes") , "TestGas_mark_a_in_skal_
110          ikke_omregnes");
111      // Fejl på hitBefore+hitAfter +"(?![\\s° -]+("+units+"))
112      // "
112      assertEquals(RecipeConverter.convert("TestGas_mark_5_
113          in_skal_ikke_omregnes") , "TestGas_mark_5in_skal_
114          ikke_omregnes");
115      // Fejl på hitUnits+hitSpaces+hitNumbers
116      assertEquals(RecipeConverter.convert("Test_Gassmarka_
117          ue_skal_ikke_omregnes") , "Test_Gassmarka_ue_skal_
118          ikke_omregnes");
119      // Fejl på hitUnits+hitSpaces+hitAfter
119      assertEquals(RecipeConverter.convert("Test_Gassmark5_
120          in_skal_ikke_omregnes") , "Test_Gassmark5_in_skal_
121          ikke_omregnes");
122      // Fejl på hitUnits+hitNumbers+hitAfter
122      assertEquals(RecipeConverter.convert("Test_Gassmark_
123          aue_skal_ikke_omregnes") , "Test_Gassmark_aue_skal_
124          ikke_omregnes");
125      // Fejl på hitUnits+hitNumbers +"(?![\\s° -]+("+units+"))
125      assertEquals(RecipeConverter.convert("Test_Gassmark_a_
126          in_skal_ikke_omregnes") , "Test_Gassmark_a_in_skal_
126          ikke_omregnes");

```

```

127          // Fejl på hitUnits+hitAfter +"(?![\\s° -]+(" + units +"))"
128          assertEquals(RecipeConverter.convert("Test_Gassmark_5
129                          in_skal_ikke_omregnes") , "Test_Gassmark_5in_skal_
130                          ikke_omregnes");
131
132          // Fejl på hitSpaces+hitNumbers+hitAfter
133          assertEquals(RecipeConverter.convert("Test_Gas_
134                          markaue_skal_ikke_omregnes") , "Test_Gas_markaue_
135                          skal_ikke_omregnes");
136
137          // Fejl på hitSpaces+hitNumbers +"(?![\\s° -]+(" + units
138          +
139          // Fejl på hitNumbers+hitAfter +"(?![\\s° -]+(" + units +"))
140          assertEquals(RecipeConverter.convert("Test_Gas_mark_
141                          ain_skal_ikke_omregnes") , "Test_Gas_mark_ain_skal_
142                          ikke_omregnes");
143
144          // Fejl på hitBefore+hitUnits+hitSpaces+hitNumbers
145          assertEquals(RecipeConverter.convert("TestGassmarka_
146                          ue_skal_ikke_omregnes") , "TestGassmarka_ue_skal_
147                          ikke_omregnes");
148
149          // Fejl på hitBefore+hitUnits+hitSpaces +"(?![\\s°
150                          -]+(" + units +"))"
151          assertEquals(RecipeConverter.convert("TestGassmark5_
152                          in_skal_ikke_omregnes") , "TestGassmark5_in_skal_
153                          ikke_omregnes");
154
155          // Fejl på hitBefore+hitUnits+hitNumbers+hitAfter

```

```

152     assertEquals(RecipeConverter.convert("TestGassmark_"
153         "aue_skal_ikke_omregnes") , "TestGassmark_aue_skal_"
154         "ikke_omregnes");
155     // Fejl på hitBefore+hitUnits+hitNumbers +"(?![\\s°"
156         "-]+("+units+"))"
157     assertEquals(RecipeConverter.convert("TestGassmark_a_"
158         "in_skal_ikke_omregnes") , "TestGassmark_a_in_skal_"
159         "ikke_omregnes");
160     // Fejl på hitBefore+hitUnits+hitAfter +"(?![\\s°-]+("+
161         "units+"))"
162     assertEquals(RecipeConverter.convert("TestGassmark_5"
163         "in_skal_ikke_omregnes") , "TestGassmark_5in_skal_"
164         "ikke_omregnes");
165     // Fejl på hitBefore+hitSpaces+hitNumbers+hitAfter
166     assertEquals(RecipeConverter.convert("TestGas_markaue_"
167         "skal_ikke_omregnes") , "TestGas_markaue_skal_ikke_"
168         "omregnes");
169     // Fejl på hitBefore+hitSpaces+hitNumbers +"(?![\\s°"
170         "-]+("+units+"))"
171     assertEquals(RecipeConverter.convert("TestGas_mark_"
172         "ain_skal_ikke_omregnes") , "TestGas_mark_ain_skal_"
173         "ikke_omregnes");
174     // Fejl på hitUnits+hitSpaces+hitNumbers+hitAfter
175     assertEquals(RecipeConverter.convert("Test_"

```

```

176           -]+("+"+units+" ))"
177     assertEquals(RecipeConverter.convert("Test_Gassmarka_
178         in_skal_ikke_omregnes") , "Test_Gassmarka_in_skal_
179             ikke_omregnes");
180
181     // Fejl på hitUnits+hitSpaces+hitAfter +"(?![\s° -]+("+
182     // units+" ))"
183     assertEquals(RecipeConverter.convert("Test_
184         Gassmark5in_skal_ikke_omregnes") , "Test_
185             Gassmark5in_skal_ikke_omregnes");
186
187     // Fejl på hitSpaces+hitNumbers+hitAfter +"(?![\s°
188     // -]+("+"+units+" ))"
189     assertEquals(RecipeConverter.convert("Test_Gas_
190         markain_skal_ikke_omregnes") , "Test_Gas_markain_
191             skal_ikke_omregnes");
192
193     // Fejl på hitBefore+hitUnits+hitSpaces+hitNumbers
194     // +"(?![\s° -]+("+"+units+" ))"
195     assertEquals(RecipeConverter.convert("TestGassmarkaue_
196         _skal_ikke_omregnes") , "TestGassmarkaue_skal_ikke_
197             _omregnes");
198
199     // Fejl på hitBefore+hitUnits+hitSpaces+hitAfter
200     // +"(?![\s° -]+("+"+units+" ))"
201     assertEquals(RecipeConverter.convert("TestGassmark5in_
202         _skal_ikke_omregnes") , "TestGassmark5in_skal_ikke_
203             _omregnes");
204
205     // Fejl på hitBefore+hitUnits+hitNumbers+hitAfter
206     // +"(?![\s° -]+("+"+units+" ))"
207     assertEquals(RecipeConverter.convert("TestGassmark_
208         ain_skal_ikke_omregnes") , "TestGassmark_ain_skal_
209             ikke_omregnes");

```

```

198     ikke_omregnes");
199     // Fejl på hitBefore+hitSpaces+hitNumbers+hitAfter
200     // +"(?![\s°-]+("+units+"))"
200     assertEquals(RecipeConverter.convert("TestGas_markain
200         skal_ikke_omregnes") , "TestGas_markain_skal_ikke
200         _omregnes");
201
202     // Fejl på hitUnits+hitSpaces+hitNumbers+hitAfter
202     // +"(?![\s°-]+("+units+"))"
203     assertEquals(RecipeConverter.convert("Test_
203         Gassmarkain_skal_ikke_omregnes") , "Test_
203         Gassmarkain_skal_ikke_omregnes");
204
205     // Fejl på hitBefore+hitUnits+hitSpaces+hitNumbers+
205     // hitAfter +"(?![\s°-]+("+units+"))"
206     assertEquals(RecipeConverter.convert("TestGassmarkain
206         skal_ikke_omregnes") , "TestGassmarkain_skal_ikke
206         _omregnes");
207
208
209 }
210
211 }
```

E.12 Case3Test.java

```

1
2 import junit.framework.TestCase;
3
4 public class Case3Test extends TestCase {
5
6     /**
7      * Metode for test af fejl på alle grænser når en unit skal
8      * konverteres.
9      * Tester om kravene hitBefore+hitNumbers+hitSpaces+hitUnits+
10     * hitAfter blir opfyldt
11     * For hver test legges ind en endring
12     */
13     public final void testNumber_Unit() {
14
15         // Test af ingen fejl
14         assertEquals(RecipeConverter.convert("Test_10_cup_
15             skal_omregnes") , "Test_28_dl_skal_omregnes");
15         // Fejl på hitBefore
```

```
16         assertEquals(RecipeConverter.convert("Test10_cup_skal_
17             _ikke_omregnes") , "Test10_cup_skal_ikke_omregnes"
18             );
19         //Fejl på hitNumbers
20         assertEquals(RecipeConverter.convert("Test_a_cup_skal_
21             _ikke_omregnes") , "Test_a_cup_skal_ikke_omregnes"
22             );
23         //Fejl på hitSpaces
24         assertEquals(RecipeConverter.convert("Test_10cup_skal_
25             _ikke_omregnes") , "Test_10cup_skal_ikke_omregnes"
26             );
27         //Fejl på hitUnits
28         assertEquals(RecipeConverter.convert("Test_10_cop_skal_
29             _ikke_omregnes") , "Test_10_cop_skal_ikke_omregnes");
30         //Fejl på hitAfter
31         assertEquals(RecipeConverter.convert("Test_10_cupskal_
32             _ikke_omregnes") , "Test_10_cupskal_ikke_omregnes");
33         //Fejl på hitBefore+hitNumbers
34         assertEquals(RecipeConverter.convert("Test_a_cup_skal_
35             _ikke_omregnes") , "Test_a_cup_skal_ikke_omregnes");
36         //Fejl på hitBefore+hitSpaces
37         assertEquals(RecipeConverter.convert("Test_acup_skal_
38             _ikke_omregnes") , "Test_acup_skal_ikke_omregnes");
39         //Fejl på hitNumbers+hitUnits
40         assertEquals(RecipeConverter.convert("Test_a_cop_skal_
41             _ikke_omregnes") , "Test_a_cop_skal_ikke_omregnes");
42         //Fejl på hitNumbers+hitAfter
43         assertEquals(RecipeConverter.convert("Test_a_cupskal_
44             _ikke_omregnes") , "Test_a_cupskal_ikke_omregnes");
45         //Fejl på hitSpaces+hitUnits
46         assertEquals(RecipeConverter.convert("Test_10cop_skal_
47             _ikke_omregnes") , "Test_10cop_skal_ikke_omregnes");
```

```

        );
// Fejl på hitSpaces+hitAfter
assertEquals(RecipeConverter.convert("Test_10cupskał_
    ikke_omregnes") , "Test_10cupskal_ikke_omregnes");
// Fejl på hitUnits+hitAfter
assertEquals(RecipeConverter.convert("Test_10_copskal_
    ikke_omregnes") , "Test_10_copskal_ikke_omregnes");
);
// Fejl på hitBefore+hitNumbers+hitSpaces
assertEquals(RecipeConverter.convert("Testacup_skal_
    ikke_omregnes") , "Testacup_skal_ikke_omregnes");
// Fejl på hitBefore+hitNumbers+hitUnits
assertEquals(RecipeConverter.convert("Testacop_skal_
    ikke_omregnes") , "Testacop_skal_ikke_omregnes");
// Fejl på hitBefore+hitNumbers+hitAfter
assertEquals(RecipeConverter.convert("Testacupskal_
    ikke_omregnes") , "Testacupskal_ikke_omregnes");
// Fejl på hitBefore+hitSpaces+hitUnits
assertEquals(RecipeConverter.convert("Test10cop_skal_
    ikke_omregnes") , "Test10cop_skal_ikke_omregnes");
// Fejl på hitBefore+hitSpaces+hitAfter
assertEquals(RecipeConverter.convert("Test10cupskal_
    ikke_omregnes") , "Test10cupskal_ikke_omregnes");
// Fejl på hitBefore+hitUnits+hitAfter
assertEquals(RecipeConverter.convert("Test10_copskal_
    ikke_omregnes") , "Test10_copskal_ikke_omregnes");
// Fejl på hitBefore+hitUnits+hitAfter
assertEquals(RecipeConverter.convert("Test10_copskal_
    ikke_omregnes") , "Test10_copskal_ikke_omregnes");
// Fejl på hitNumbers+hitSpaces+hitUnits
assertEquals(RecipeConverter.convert("Test_acop_skal_
    ikke_omregnes") , "Test_acop_skal_ikke_omregnes");
// Fejl på hitNumbers+hitSpaces+hitAfter
assertEquals(RecipeConverter.convert("Test_acupskal_
    ikke_omregnes") , "Test_acupskal_ikke_omregnes");
// Fejl på hitNumbers+hitUnits+hitAfter
assertEquals(RecipeConverter.convert("Test_a_copskal_
    ikke_omregnes") , "Test_a_copskal_ikke_omregnes");
// Fejl på hitSpaces+hitUnits+hitAfter
assertEquals(RecipeConverter.convert("Test_10copskał_
    ikke_omregnes") , "Test_10copskał_ikke_omregnes");
// Fejl på hitBefore+hitNumbers+hitSpaces+hitUnits
assertEquals(RecipeConverter.convert("Testacop_skal_
    ikke_omregnes") , "Testacop_skal_ikke_omregnes");
// Fejl på hitBefore+hitNumbers+hitSpaces+hitAfter
assertEquals(RecipeConverter.convert("Testacupskal_
    ikke_omregnes") , "Testacupskal_ikke_omregnes");
// Fejl på hitBefore+hitNumbers+hitUnits+hitAfter

```

```
70         assertEquals(RecipeConverter.convert("Testacopskal_"
71             "ikke_omregnes") , "Testacopskal_ikke_omregnes");
72     // Fejl på hitBefore+hitSpaces+hitUnits+hitAfter
73     assertEquals(RecipeConverter.convert("Test10copskal_"
74             "ikke_omregnes") , "Test10copskal_ikke_omregnes");
75     // Fejl på hitNumbers+hitSpaces+hitUnits+hitAfter
76     assertEquals(RecipeConverter.convert("Test_acopskal_"
77             "ikke_omregnes") , "Test_acopskal_ikke_omregnes");
78     // Fejl på hitBefore+hitNumbers+hitSpaces+hitUnits+
79     hitAfter
    assertEquals(RecipeConverter.convert("Testacopskal_"
        "ikke_omregnes") , "Testacopskal_ikke_omregnes");
}
}
```

Bilag F

Testskemaer Blackbox og whitebox