Kassa System AB

Introduktion till testing av it-system 2015-10-28

Grupp 7:
Jacob Nienhuysen jani1342
Erik Holmström erho4567
Paulina Palmé papa6605
Leo Oxfält leox9811

Introduktion

Den projektuppgift som valdes var Kvitton och Rabatter som går ut på att gruppen ska implementera klasser som skulle kunna användas i ett kassasystem för att representera kvitton och rabatter.

Verktyg:

Versionshanterare:

Vi valde att använda oss av GitHub:

https://github.com/jacobnienhuysen/INTEGrupp7.git

Byggserver:

Eclipse

NetBeans

Byggscript

ANT

Enhetstestramverk:

JUnit

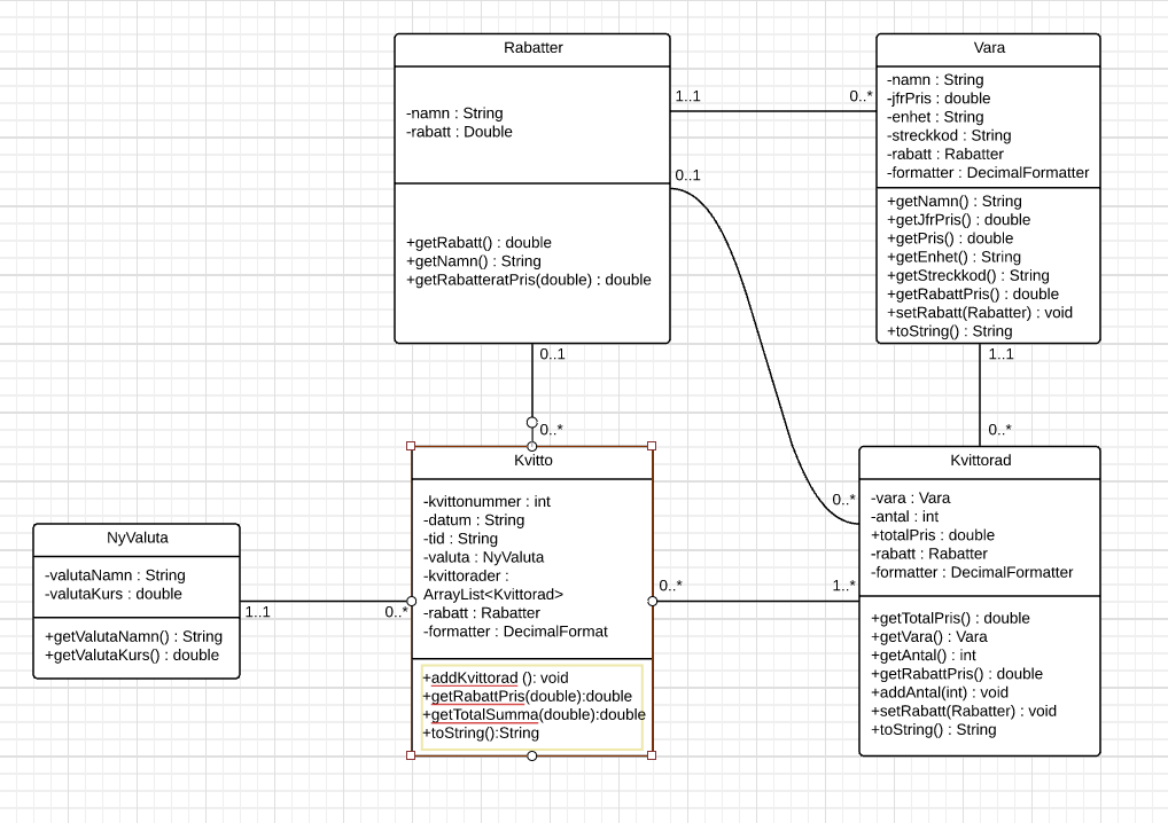
Övriga verktyg

Notepad ++

Office paketet

Facebook

Slutlig design



Testdriven utveckling – process

Efter att ha ritat upp ett klassdiagram över systemet valde vi att dela upp klasserna mellan oss. Därefter skrev vi ett JUnit test och sedan skrev vi den minsta mängden nödvändig kod för att testet skulle lyckas. Detta gjorde vi fram tills att klassen var klar och då gick vi vidare till nästa klass.

Exempel på TDD från systemet (Klass Rabatter):

```
@Test
public void testRabatt(){
    Rabatter r = new Rabatter("TestR", 10);
    assertEquals(0.9, r.getRabatt(), 0.0);
}

@Test
public void testNamn(){
    Rabatter r = new Rabatter("TestR", 10);
    assertEquals("TestR", r.getNamn());
}

@Test
public void testReturneraRabateratPris(){
    Rabatter r = new Rabatter("TestR", 10);
    assertEquals(90.0, r.getRabatteratPris(100.0), 0.0);
}
@Test(expected = IllegalArgumentException.class)
public void testRabattForStor(){
    Rabatter r = new Rabatter("TestR", 110);
}
@Test(expected = IllegalArgumentException.class)
public void testRabattForLiten(){
    Rabatter r = new Rabatter("TestR", -1);
}
```

```
public class Rabatter {
    private String namn;
    private Double rabatt;
    public Rabatter(String n, int r){
        if(n.length()<1 || !n.matches(".*[a-zA-Z0-9Ã¥Ã¤Ã¶Ã…Ã„Ã–]+.*"))
            throw new IllegalArgumentException("Ogiltigt namn pÃ¥ vara");

        if(r > 100 || r < 0)
            throw new IllegalArgumentException("MÃ¥ste vara mellan 0 och 100");

        namn = n;
        rabatt = (double)(100-r)/100;
    }
    public Double getRabatt(){
        return rabatt;
    }
    public String getNamn(){
        return namn;
    }
    public Double getRabatteratPris(Double pris){
        return pris*rabatt;
    }
}
```

Testdriven utveckling – erfarenheter

Vi upplevde att projektet i sig är så pass litet att vi kände att det eventuellt hade varit enklare att skapa systemet utan TDD. Men vi förstår det praktiska med att använda sig av TDD i större projekt som är mer komplexa. Eftersom man har ett program som kompilerar vet man att man har kod som fungerar och inte behöver testas.

En annan erfarenhet som vi fått av att använda oss av TDD är att när man hittar fel i koden och gör en förändring, så finns testen där för att ge en direkt feedback på om man förstört något annat när man gjorde sin förändring.

Ekvivalensklassuppdelning – Hela systemet

Då flera av klasserna i systemet kräver input i form av objekt av andra klasser i systemet, valde vi att utföra ekvivalensklassuppdelning på alla delar samtidigt. Exempelvis går det inte att skapa ett kvitto utan att man först skapat både ett valutaobjekt och en kvittorad som i sin tur kräver att man matar in ett objekt av klassen Vara.

Det ledde även till att flera av de valida ekvivalensklasserna överlappas i testfallen, för att testerna skulle generera godkända resultat.

Ekvivalensklasser – Hela systemet

ID	Ekvivalensklass	Valid?	Anteckningar
V1	Vara pris =>1	x	En vara måste kosta något
V2	Vara pris < 0		En vara kan inte ett pris som är minus
V3	Vara enhet KG	x	
V4	Vara enhet Styck	x	
V5	Vara enhet Liter	x	
V6	Vara enhet deciliter		
V7	Vara enhet mililiter		
V8	vara namn = A–Ö	x	
V9	vara namn != A–Ö		
V10	vara enhet Gram	x	
V11	vara enhet Hg	x	
V12	Streckkod <= 12 siffror		
V13	Streckkod >= 14 siffror		
V14	Steckkod = 13 siffror	x	
V16	Streckkod != siffror		

ID	Ekvivalensklass	Valid?	Anteckningar
VA1	valuta namn dollar	x	
VA2	valuta namn euro	x	
VA3	valuta namn sek	x	
VA4	valuta namn != dollar, euro, sek		
VA5	valuta kurs < 0		
VA6	valuta kurs > 0	x	

ID	Ekvivalensklass	Valid?	Anteckningar
E1	kvitto nummer < 0		
E2	kvitto nummer > 0	x	
E3	Tid = nuvarande tid	x	
E4	Tid != nuvarande tid		
E5	Totalpris > 0	x	
E6	Totalpris < 0		
E7	Antal > 0	x	
E8	Antal <= 0		
E9	kvittorader<1		
E10	kvittorader>=1	x	

ID	Ekvivalensklass	Valid?	Anteckningar
KR1	antal av vara <1		
KR2	antal av vara >=1	x	

ID	Ekvivalensklass	Valid?	Anteckningar
R1	Rabatt namn = A–Ö	x	
R2	Rabatt namn != A–Ö		
R3	Rabatt värde <= 0%		
R4	Rabatt värde > 0%	x	
R6	Rabatt värde >= 100%		

Testfall – Hela systemet

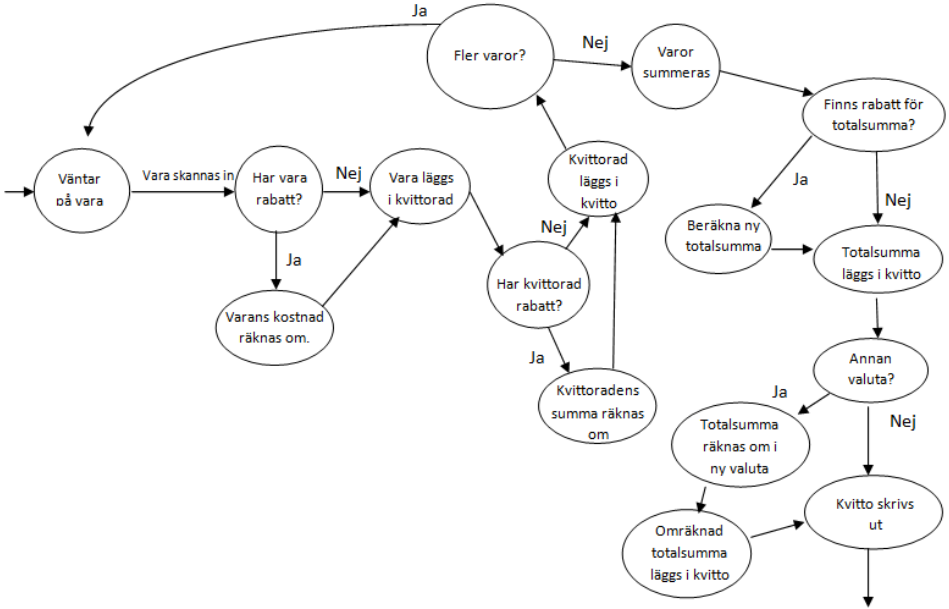
ID	Testfall	Input	Förvänta	Ekvivalensklass
T1	Vara pris>=1,Valuta namn dollar, Testa enhet KG, Testa namn, Streckkod=13 siffror Valuta kurs >0, Kvittonummer > 0 siffror, Tid=nuvarande tid, Totalpris>0, Antal>0, Rabattnamn, Rabattvärde>0	12, dollar, Kg, Äpplen, 1234567891234, 8.51, 123, timestamp, 24, 13, Pensionär, 10%	ok	V1, V3, V8, V14, VA1, VA6, E2, E3, E5, E7, R1, R4, E10, KR2
T2	Vara pris>=1, Vara enhet styck, Varunamn A-Ö, Streckkod=13 siffror Valuta namn euro, Valuta kurs>0, Kvittonummer> 0siffror, Tid=nuvarande tid, Antal>0	5, styck, Sudd, 1234567890123, euro, 9.39, 12345, timestamp, 23,	ok	V1, V4, V8, V15, VA2, VA6, E2, E3, E5, E7, E10, KR2
T3	Vara pris>=1, Vara enhet liter, Varunamn A-Ö, Streckkod=13 siffror Valutanamn sek, Valuta kurs>0, Kvittonummer >0 siffror, Tid=nuvarande tid, Totalpris>0, Antal>0,	16, liter, Mjölk, 9876543210987, sek, 1, 567, timestamp, 27, 4	ok	V1, V5, V8, V14, VA3, VA6, E2, E3, E5, E7, E10,KR2
T4	Vara pris>=1, Varunamn A-Ö, Vara enhet gram, Streckkod=13 siffror Valuta namn dollar, Valuta kurs>0, Kvittonummer>0 siffror, Tid= numvarande tid, Totalpris>0, Antal>0,	18, saffran, gram, 2345678901111, dollar, 9.13, 987, timestamp, 19.95, 1,	ok	V1, V8, V10, V15, VA1, VA6, E2, E3, E5, E7, E10, KR2
T5	Vara pris>=1, Varunamn A-Ö, Vara enhet hekto, Streckkod =13 siffror, Valutanamn euro, Valuta kurs>0, Kvittonummer>0siffror, Tid=nuvarande tid, Totalpris>0, Antal>0	14, Lösgodis, hekto, 3456789012345, euro, 9.39, 45678, timestamp, 50, 1	ok	V1, V8, V11, V14, VA2, VA6, E2, E3, E5, E7, E10, KR2
T6	Vara <=0	0, Lösgodis, hekto, 3456789012345, euro, 9.39, 45678, timestamp, 50, 1	INTE ok	V2
T7	Vara enhet deciliter	16, deciliter, grillkol 9876543210987, sek, 1, 567, timestamp, 27, 4	INTE ok	V6
T8	Vara enhet milliliter	16, milliliter, grillkol 9876543210987, sek, 1, 567, timestamp, 27, 4	INTE ok	V7
T9	Vara namns != A-Ö	12, dollar, Kg, @pplen, 1234567891234, 8.51, 123, timestamp, 24, 13, Pensionär, 10%	INTE ok	V9
T10	Streckkod<=12 siffror	14, Lösgodis, hekto, 3456, euro, 9.39, 45678, timestamp, 50, 1	INTE ok	V12
T11	Streckkod >= 14 siffror	14, Lösgodis, hekto, 3456789012345123, euro, 9.39, 45678, timestamp, 50, 1	INTE ok	V13
T12	Streckkod != siffror	14, Lösgodis, hekto, 345678BA12345, euro, 9.39, 45678, timestamp, 50, 1	INTE ok	V16
T13	valuta namn != dollar, euro, sek	18, saffran, gram, 2345678901111, nok, 9.13, 987, timestamp, 19.95, 1,	INTE ok	VA4
T14	valuta kurs < 0	18, saffran, gram, 2345678901111, dollar, -0.13, 987, timestamp, 19.95, 1,	INTE ok	VA5
T15	kvitto nummer < 0	12, dollar, Kg, Äpplen, 1234567891234, 8.51, inget kvittonummer, timestamp, 24, 13, Pensionär, 10%	INTE ok	E1
T16	Tid != nuvarande tid	5, styck, Sudd, 1234567890123, euro, 9.39, 12345, 23:52, 23,	INTE ok	E4
T17	Totalpris < 0	12, dollar, Kg, Äpplen, 1234567891234, 8.51, 123, timestamp, -24, 13, Pensionär, 10%	INTE ok	E6
T18	Antal <= 0	18, saffran, gram, 2345678901111, dollar, 9.13, 987, timestamp, 19.95, -1,	INTE ok	E8
T19	Rabatt namn != A-Ö	12, dollar, Kg, Äpplen, 1234567891234, 8.51, 123, timestamp, 24, 13, Pension@r, 10%	INTE ok	R2
T20	Rabatt värde <= 0%	12, dollar, Kg, Äpplen, 1234567891234, 8.51, 123, timestamp, 24, 13, Pensionär, -10%	INTE ok	R3
T21	Rabatt värde >= 100%	12, dollar, Kg, Äpplen, 1234567891234, 8.51, 123, timestamp, 24, 13, Pensionär, 110%	INTE ok	R6
T22	Kvittorader<1	Kvitto(987, null)	INTE ok	E9
T23	antal av vara <1	Finns igen möjlig	INTE ok	KR1

Testmatris – Hela systemet

		Ekvivalensklasser																																									
		V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	VA1	VA2	VA3	VA4	VA5	VA6	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	KR1	KR2	R1	R2	R3	R4	R6
Testfall	T1	x		x					x						x			x					x		x	x		x		x			x		x	x			x	
	T2	x			x			x								x			x				x		x	x		x		x			x		x					
	T3	x				x			x						x					x			x		x	x		x		x			x		x					
	T4	x							x		x					x		x					x		x	x		x		x			x		x					
	T5	x							x			x			x				x				x		x	x		x		x			x		x					
	T6		x																																						
	T7						x																																		
	T8							x																																	
	T9									x																															
	T10												x																												
	T11													x																											
	T12																x																								
	T13																				x																			
	T14																					x																		
	T15																							x																
	T16																										x													
	T17																												x										
	T18																														x								
	T19																																				x			
	T20																																					x		
	T21																																							x
	T22																															x								
	T23																																	x						

Tillståndsbaserad testning

Eftersom vi valde att göra ekvivalensklasser för hela systemet valde vi att även gör en tillståndsmaskin för hela systemet. Vi har valt att använda oss av täckningskriteriumet statement coverage. Valet av täckningsgrad baseras på att vi dels vill täcka alla övergångar(bransch coverage) vilket gör att vi samtidigt täcker alla tillstånd och uppnår då statement coverage.



Testfall för tillståndsbaserad testning

ID	Beskrivning	Täckta tillstånd	Täckta övergångar
1	1, Skanna in vara 2, Vara har rabatt 3, Kvittorad har rabatt 4, Inga fler varor 5, Rabatt för totalsumma 6, Annan valuta vald	Väntar på vara Har vara rabatt? Varans kostnad räknas om Vara läggs i kvittorad Har kvittorad rabatt? Kvittoradens summa räknas om Kvittorad läggs i kvitto Fler varor? Varor summeras Finns rabatt för totalsumma? Beräkna ny totalsumma Totalsumma läggs i kvitto Annan valuta? Totalsumma räknas om i ny valuta Omräknad totalsumma läggs i kvitto Kvitto skrivs ut	Vara skannas in Ja Ja Nej Ja Ja
2	1, Skanna in vara 2, Vara har inte rabatt 3, Kvittorad har inte rabatt 4, Fler varor 5, Skanna in vara 6, Vara har inte rabatt 7, Kvittorad har inte rabatt 8, Inga fler varor 9, Ingen rabatt för totalsumma 10, Ingen annan valuta vald	Vänta på vara Har vara rabatt? Vara läggs i kvittorad Har kvittorad rabatt? Kvittorad läggs i kvitto Fler varor? Skanna in vara Har vara rabatt? Vara läggs i kvittorad Har kvittorad rabatt? Kvittorad läggs i kvitto Fler varor? Varor summeras Finns rabatt för totalsumma? Totalsumma läggs i kvitto Annan valuta? Kvitto skrivs ut	Vara skannas in Nej Nej Ja Vara skannas in Nej Nej Nej Nej Nej

Granskning

Vi valde att göra en formell granskning av hela systemet. Eftersom projektet är så pass litet valde vi att inte genomföra granskningen fullt så formellt som den beskrivs i boken, utan genom en enklare inspektion av varandras skrivna klasser. Vid funderingar frågades författaren av klassens kod om hur författaren tänkt och varför. För att underlätta granskningen och se till att alla delar vart granskade valde vi att använda oss av checklistan från workshopen: (https://ilearn2.dsv.su.se/pluginfile.php/51707/mod_resource/content/0/checklist.pdf).
Vi valde att använda denna checklista eftersom att vi använt den tidigare under workshopen och upplevde att den var enklare att följa en de som togs upp under föreläsningen.

Eftersom att koden inte är så stor och när vi kodat har vi varit väldigt noga upptäcktes inte så många fel under granskningen. De fel som kom upp under granskningen dokumenterades i en tabell som kan ses nedan.

Granskningsrapport

Påträffat fel	Beskrivning	Allvarlighet (1-5)
ÅÄÖ i kod	Visas inte korrekt pågrund av skillnaden i datorernas språk inställningar	2 I metod och klassnamn ställer det till besvär. Annars rent estetiska problem
double deklarerat med stort D	I deklarationen av variabeln double är den skriven med stor D	1 Inte konsekvent med resten av koden.

Granskning – erfarenheter

Efter att ha granskat varandras kod insåg vi att det är lätt att missa fel eftersom att man ofta har väldigt stor tilltro till sig själv och sin egen kod. Vi upplevde även att det ibland kunde kännas svårt att rapportera ett fel som man har upptäckt eftersom att det kan bli dålig stämning i gruppen. Men med vetskapen i ryggen att det är viktigt att rapportera alla fel som man hittar även fast författaren av koden kan ta illa vid sig upplevde vi att det kändes lättare.

En viktig del av hur man rapporterar fel är hur man lägger fram det och vi märkte tidigt att det bästa sättet var att öppet säga och dokumentera ett fel utan att ifrågasätta vem som gjort felet eller varför.

När vi granskade efter att vi hade kört FindBugs insåg vi att det är lätt att missa mindre uppenbara fel, som till exempel variabler som inte används eller metoder som anropar variabler som riskerar att inte ha blivit initierade.

Under granskning av någon annans kod är det lätt att man hakar upp sig på hur andra programmerare strukturerar sin kod eftersom det skiljer sig ifrån vad man själv tycker är korrekt.

Kodkritiksystem

Som kodkritiksystem valde vi att använda oss av FindBugs på hela systemet. Fel hittades bara i klassen Kvittorad och klassen Vara.

Fel som hittats före den formella granskningen:

Påträffat fel	Beskrivning	Allvarlighet (1-5)
I klassen Vara: Use of non –localized String.toUpperCase() or String.to.LowerCase()	När vi inte angav någon språkkodning reagerade den eftersom att det finns en risk att den inte konverterar internationella tecknen rätt.	3
I klassen Kvittorad: Unused public or protected filed…	Variabeln totalpris används inte men eftersom att den är deklarerad i public misstänker den att den kan komma att användas utifrån.	1 Egentligen ingen större fara att den är public eftersom den inte används i koden överhuvudtaget.
I klassen Kvittorad och i klassen Vara: Vara rabatt is not initialized	Variablen rabatt initieras inte i konstruktorn, men kan ändå anropas i en metod.	2 Eftersom rabatt inte initieras men ändå kan anropas i metoden utan någon kontroll finns risken att man får null i returvärde.

Kodkritiksystem

Fel som hittats med FindBugs efter den formella granskningen och rättningen:

När vi jämförde vad vi hittade under granskningen med resultatet från FindBugs-körningen insåg vi att vi missade ett par fel som FindBugs upptäckte. Detta skulle kunna vara ett tecken på att vi var dåliga på granska, men om man kollar på de fel som FindBugs faktiskt upptäckte kan man konstatera att det var fel som är svåra för en granskare att upptäcka när koden gås igenom.

Statiska mått

Vi valde att utföra statiska mått med hjälp av programmet Metrix 1.3.6 på hela systemet inklusive testklasserna. Resultatet kan ses i tabellen nedan

Statiskt mått	Resultat	Diskussion
Antal rader kod	233st	
Flest rader kod	Klassen kvitto (81st)	
Antal klasser	5st	
McCabe Medel	1,722st	
McCabe Max i klassen Vara	11st	Väldigt intressant att det fanns så många vägar i klassen Vara vilket visade på att klassen är komplexare än vad vi trodde.
Attribut totalt	22st	Intressant att se att måtten stämmer överens med det klassdiagram som vi ritade upp i designstadiet av programmet. Detta visar på att vi har byggt det system som vi från början tänkt bygga.
Attribut i klassen Kvitto	7st	
Attribut i klassen Vara	6st	
Attribut i klassen Kvittorad	5st	
Attribut i klassen Rabatter	2st	
Attribut i klassen Valuta	2st	
Afferent Coupling	20	Antal klasser som är beroende av andra klasser i samma paket. Detta är intressant eftersom det är 5 klasser men hela 20 kopplingar vilket tyder på att dessa klasser är väldigt beroende av varandra för att fungera.

Täckningsgrad

Vi valde att använda oss av programmet EclEmma för att mäta vår täckningsgrad. Våra testfall uppnår 100% täckningsgrad för själva produktionskoden men inte på vår testkod. Detta beror på att testfallen förväntar sig ett fel och när de får ett fel i anropet avbryts testet. När systemet avbrutit ett testfall räknas det som att det inte är kört även fast det är det.

Profiler

Vi använde oss av NetBeans för att testa koden med profiler. Vi använde den inbyggda standardprofilern och testade både hur mycket minne och processorkraft som gick åt vid körning. Därefter gjorde vi en analys av resultaten som vi fick fram.

Vid körningen av minnes-profilern kunde vi se att systemet främst arbetade med objekt av typen int och char. Att integers användes i stor utsträckning var ganska väntat, eftersom vi nästan uteslutande gör matematiska uträkningar. Chars används lite mer till följd av att vi i klassen Vara anropar metoden toLowerCase, för att försäkra att enheten alltid lagras i gemener även om användaren råkar mata in versaler.

Ur CPU-profilern fick vi nästan inte ut några värden alls eftersom projektet är så pass litet. Man skulle kanske kunna tvinga fram ett resultat som vara större om man skrev ett längre testfall som gjorde mer och om koden innehöll fler komplicerade och/eller minneskrävande beräkningar.

Byggscript

Vi valde att använda oss av programmet ANT för att bygga vårt byggscript. Slutliga utformning:

```
<?xml version="1.0"?>
<project default="main" basedir="." name="INTEProjekt">
    <property name="src.dir" value="src"/>
    <property name="build.dir" value="build"/>
    <property name="classes.dir" value="${build.dir}/classes"/>
    <property name="jar.dir" value="${build.dir}/jar"/>
    <property name="main-class" value="inteprojekt.Kvitto"/>
    <property name="lib.dir" value="lib"/>
    <property name="report.dir" value="${build.dir}/junitreport"/>
  - <path id="classpath">
        <fileset dir="${lib.dir}" includes="**/*.jar"/>
    </path>
    <path id="application" location="${jar.dir}/${ant.project.name}.jar"/>
  - <target name="clean">
        <delete dir="${build.dir}"/>
    </target>
  - <target name="compile">
        <mkdir dir="${classes.dir}"/>
        <javac includeantruntime="false" classpathref="classpath" destdir="${classes.dir}" srcdir="${src.dir}"/>
      - <copy todir="${classes.dir}">
            <fileset dir="${src.dir}" excludes="**/*.java"/>
        </copy>
    </target>
  - <target name="jar" depends="compile">
        <mkdir dir="${jar.dir}"/>
      - <jar basedir="${classes.dir}" destfile="${jar.dir}/${ant.project.name}.jar">
          - <manifest>
                <attribute name="Main-Class" value="${main-class}"/>
            </manifest>
        </jar>
    </target>
  - <target name="run" depends="jar">
      - <java classname="${main-class}" fork="true">
          - <classpath>
                <path refid="classpath"/>
                <path refid="application"/>
            </classpath>
        </java>
    </target>
  - <target name="junit" depends="jar">
        <mkdir dir="${report.dir}"/>
      - <junit haltonfailure="yes" printsummary="yes">
          - <classpath>
                <path refid="classpath"/>
                <path refid="application"/>
            </classpath>
            <formatter type="plain"/>
            <formatter type="xml"/>
          - <batchtest todir="${report.dir}" fork="yes">
                <fileset dir="${src.dir}" includes="**/*Test*.java"/>
            </batchtest>
        </junit>
    </target>
  - <target name="junitreport">
      - <junitreport todir="${report.dir}">
            <fileset dir="${report.dir}" includes="TEST-*.xml"/>
            <report todir="${report.dir}"/>
        </junitreport>
    </target>
    <target name="clean-build" depends="clean,jar"/>
    <target name="main" depends="clean,run"/>
</project>
```

Övrigt

Internationalisering

Vi upptäckte att vårt kassasystem hanterade punkt och komma i pris på olika sätt beroende på om det kördes på en dator med operativsystemet inställt på svenska eller engelska. Detta eftersom att java hanterar tecknen olika. Vi valde att bortse från detta i vårt system eftersom att det skulle bli för omfattande för den här kursen.

ANT tycker inte om svenska

Vi upptäckte att ANT inte alls tycker om Å, Ä, Ö vilket blev ett litet problem eftersom vissa klasser hade dessa svenska tecken i sig. Följden av att ANT inte tyckte om svenska tecken vart att ANT inte kunde köras.