Projektnamn

Introduktion till testing av it-system 2015-10-20

|  |  |
| --- | --- |
| Jacob Nienhuysen | jani1342 |
| Erik Holmström | erho4567 |
| Paulina Palmé | papa6605 |
| Diana Delta | leox9811 |

Obs! Ändra inte på formatet, det ska vara liggande A5.

# 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

# Slutlig design

En övergripande modell över systemet. Lämpligt format är ett eller flera klassdiagram, plus eventuella andra modeller som behövs för att förstå hur systemet är uppbyggt. Diagrammen ska vara läsbara. Det är dock fullständigt okej att de är detaljerade, bara det går att zooma in ordentligt på dem. Ett tips är att börja med ett översiktligt diagram som inte innehåller mer än paket och klassnamn, och att sedan lägga till mer detaljerade diagram efter det.

# Testdriven utveckling – process

*En översikt över hur ni tillämpat TDD med exempel från olika personer och olika faser i projektet. Om ni har använt versionshanteringssystemet ordentligt bör all information som efterfrågas här finnas i det. Tänk på att kodexemplen ska vara läsbara.*

Efter att ha ritat upp ett klassdiagram över systemet valde vi att dela upp klasserna mellan oss och börja utveckla klasserna separat med hjälp av TDD och på så vis börja skapa enheter.

# Testdriven utveckling – erfarenheter

*En diskussion om vilka era erfarenheter ni dragit av att tillämpa TDD. Det finns inget rätt eller fel här. Enda sättet att bli underkända är att bara fuska över punkten och säga något pliktskyldigt.*

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

# Ekvivalensklassuppdelning – namn på del

*En kort presentation av vad ni valt ut för att tillämpa ekvivalensklassuppdelning på. Ni ska kort motivera valet, och ge tillräckligt med information för att det ska gå att bedöma er. Detta avsnitt och de tre föjande (till och med testmatrisen) ska finnas för samtliga delar ni tillämpat ekvivalensklassuppdelning på.*

* Inputen var tvungen att innehålla allt runt omkring för att då ett fungerande testfall

# Ekvivalensklasser – namn på del

*Samtliga ekvivalensklasser för denna del presenterade på ett tydligt sätt.*

# Testfall – namn på del

*Testfallen som ni fått fram från ekvivalensklasserna. Observera att vi inte vill ha någon kod här, utan bara en tydlig presentation av testfallen i någon lämplig tabellform.*

# Testmatris – namn på del

*En testmatris som visar sambandet mellan ekvivalensklasserna och testfallen för denna del.*

# Tillståndsbaserad testning

*En kort presentation av vad ni valt ut för att tillämpa tillståndsbaserad testning på och vilket täckningskriterium ni valt att använda er av. Ni ska kort motivera valen, och ge tillräckligt med information för att det ska gå att bedöma er. Glöm inte att ta med själva modellen.*

# Testfall för tillståndsbaserad testning

Testfallen som ni fått fram från tillståndsmaskinen. Observera att vi inte vill ha någon kod här, utan bara en tydlig presentation av testfallen i någon lämplig tabellform. Det ska enkelt gå att mappa testfallen till tillståndsmaskinen.

# Granskning

*En kort presentation av den del av koden ni valt ut för att göra en formell granskning av och processen ni använt er av inklusive eventuella checklistor, scenarier, edyl. Ni ska kort motivera valen, och ge tillräckligt med information för att det ska gå att bedöma er.*

# Granskningsrapport

*En lista över de påträffade felen och hur pass allvarliga ni bedömer dem.*

# Granskning – erfarenheter

*En diskussion om vilka era erfarenheter ni dragit av att tillämpa granskning. Det finns inget rätt eller fel här. Enda sättet att bli underkända är att bara fuska över punkten och bara säga något pliktskyldigt. Ni förväntas förhålla er till såväl kursboken som utdelat material och IEEE Std 1028.*

# Kodkritiksystem

*En presentation av de problem som hittats med hjälp av verktyg för statisk analys och en diskussion av dem enligt anvisningarna. Det räcker alltså inte med att bara lista problemen, ni måste förhålla er till dem också. Tänk också på att ni ska göra detta både på koden som den såg ut före granskningen och på koden efter att ni rättat det som kommit fram under granskningen*.

# Statiska mått

*En presentation och diskussion kring ett antal lämpliga statiska mått på koden. Att vi inte specificerar exakt vilka mått som ska tas upp beror på att olika verktyg har olika uppsättningar, men vi förväntar oss fler och mer intressanta mått än bara rena storleksmått som LOC, #klasser, #metoder, etc. Även här är det viktigt att förhållas sig till måtten, inte bara lista dem.*

# Täckningsgrad

*En översikt över vilken täckningsgrad era testfall uppnått. Denna kan antagligen tas rakt av från verktyget ni använt för att mäta den. Om ni inte uppnått fullständig täckning så ska detta förklaras och motiveras.*

# Profiler

*En kort presentation av hur ni gått tillväga för att testa koden med en profiler och vilka resultat ni fick fram.*

# Byggscript

*Byggscriptets första (seriösa) version, och den slutliga.*

# Övrigt

*Här kan ni ta upp övrigt av relevans för bedömningen av ert arbete. Om avsnittet inte behövs kan det plockas bort.*

**Internationalisering**  
Vi upptäckte att vårt kassasystem hanterade punkt och komma i pris på olika sätt beroende på om det kördes på en dator svensk eller engelsk dator. Detta eftersom att java hanterar teckna olika. Vi valde att bortse från detta i vårt system eftersom att det skulle bli för omfattande för den här kursen.