TDP007 - Seminarium 3

Hur har ni gått tillväga för att lära er Ruby?

Föreläsningarna har hjälpt en hel del gällande detta avsnitt av kursen. Rubys dokumentation har hjälpt oss en hel del, samt forum såsom Stackoverflow och geeksforgeeks.

Vad finns det för nya konstruktioner i Ruby som ni inte sett förut?

Instance_eval är helt nytt för detta seminarium. Intressant att det går att jobba på detta sättet, det är helt nytt för oss.

Vad finns det för konstruktioner som ni känner igen men som ser lite annorlunda ut?

attr_reader är som att skapa getters fast man kan få variablerna direkt.

Finns det något som ni irriterar er på eller tycker om i Ruby?

Allting är ännu en gång lite "för" enkelt, saker som inte hade fungerat i andra språk fungerar i Ruby utan större ansträngning.

Uppgift 1

Hur har ni tolkat uppgiften?

Vi har tolkat uppgiften som att vi ska göra ett program som tolkar ruby-språk i en annan fil. I just denna uppgift ska den filen innehålla olika poäng som tillhör olika egenskaper, såsom kön, ålder, postnummer osv. Programmet ska sedan kunna skapa ett personobjekt med egenskaper som motsvarar egenskaperna i filen, och vi ska få ut en totalpoäng för personen utifrån dessa egenskaper. I detta fall används poängen för att räkna ut en personlig försäkringspremie för bilar.

Vad var svårt eller lätt med uppgiften?

Det var svårt att göra uppgiften generell. Vi lyckades inte komma hela vägen dit vi ville med den tidsramen vi hade att jobba med, men för att göra den mer generell skulle vi vilja ha att alla ranges i Policy.rb hade samma name istället för years och span.

För testerna gick vi in med strategin att göra ett test för varje regel för att se att det faktiskt gjorde någon skillnad när vi testade olika märken mot varandra till exempel. Något vi kunde tänkt på var att skapa en person med "fel" egenskaper och se hur det hade fungerat. Vi hade också kunnat testa method_missing.

Uppgift 2

Hur har ni tolkat uppgiften?

Vi har tolkat uppgiften som att vi ska kolla på hur rdparse.rb fungerar med exemplet som innehåller klassen Diceroller, och i en annan fil skapa en klass som hanterar logiska uttryck såsom and, or, not, true och false med hjälp av rdparsern. De variabler vi kan kolla dessa logiska uttryck på bestämmer vi själva. Vi valde att hantera strängar och integers.

Vad var svårt eller lätt med uppgiften?

Att förstå själva uppgiften var det svåraste eftersom det inte liknar något vi gjort sedan tidigare. Vi trodde först att uppgiften gick ut på något liknande som uppgift 1, att vi skulle skapa regler för den redan befintliga klassen, och inte att vi i själva verket skulle skapa en helt ny klass som använder sig av helt egen BNF-grammatik och rdparsern.

När vi väl insett hur vi skulle göra uppgiften var det inte jättesvårt att kolla på hur Diceroller klassen var uppbyggd och följa de instruktioner om hur varje del av uppgiften skulle vara uppbyggd från uppgiftsbeskrivningen. Det enda vi hade missat som vi inte förstod var att vi i token var tvungen att lägga till ett regex-mönster som kan matcha bokstäver då det sedan tidigare endast funnits ‘\s’, ‘\d’ och ‘.’.

Det tog ett tag innan vi förstod att vår variabeltilldelning inte fungerade heller. Varje variabel kördes en gång men hade ingenstans att lagras, så satte man a till true så lagrades det aldrig. Vi löste detta genom att följa exemplet i expression.rb och skapa en hash där vi sparar variabler i.

Vi hade också lite problem att komma på tester för uppgiften då man kör hela programmet i interaktivt läge. Vi löste detta genom att ta bort rekursionen i funktionen roll och att den istället skulle returnera det funktionen printar ut. Detta ger att vi kan testa resultatet med hjälp av assert_equal som vanligt. Efter detta var gjort var det inga större problem att testa de olika sakerna, dock hade vi önskat att man inte behövde skriva in alla tester själv i terminalen utan att det skedde direkt. Detta löste vi genom att skapa en “roll_test” funktion och behålla “roll” som den var för att behålla programmets ursprungsfunktionalitet men att också kunna testa det utan för mycket strul. Det som skiljer funktionerna åt är att roll_test endast returnerar det roll printar ut, medans roll printar ut resultatet av funktionen och kör sedan sig själv igen rekursivt.