Seminarie 4 | Utvecklarblogg

Hur har ni gått tillväga för att lära er Ruby?
Föreläsningarna och labassistenterna har hjälp till väldigt mycket för just detta. Väldigt specfik uppgift som var relativt svårt att hitta saker på internet om.

Uppgift 1
Hur har ni tolkat uppgiften?

Vi har tolkat denna uppgift som att vi ska redigera ett existerande system(nätverk) så att det ska fungera med funktionerna multiplier och adder, vilket inkluderar subtraktion och division. Detta ska sedan fungera med att konvertera celsius till fahrenheit samt åt andra hållet. Detta ska göras med hjälp av connectors och constantconnectors i nätverket. Vi ska sedan skriva enhetstester för detta.

Vad var svårt eller lätt med uppgiften?

Det tog ett tag att förstå vad hela filen innebar, och vilket problem det var vi skulle lösa med hjälp av filen. Hela uppgiften i sig var något väldigt nytt så det svåraste var väl egentligen att förstå vad och hur det skulle göras. Sedan var det svårt att ta reda på hur vi kunde lägga till subtraktion och division i funktionerna för adder och multiplier. Vi kollade hur de befintliga funktionerna såg ut, och omvände sedan funktionaliteten hos adder och multiplier för att få subtraktion och division att fungera för önskat resultat.

Eftersom att uppgift 1 binder in med uppgift 2 så behövde vi efterhand också redigera en del i uppgift1 för att få uppgift 2 att fungera optimalt. I början hade adder och multiplier all funktionalitet för addition, subtraktion respektive multiplikation och division, men för att få uppgift2 att fungera optimalt lade vi till egna klasser för subtraktion och division. Dock märkte vi att adder och multiplier fortfarande behövde ha funktionalitet för subtraktion respektive division för att programmet fortfarande skulle fungera.

Uppgift 2
Hur har ni tolkat uppgiften?

Vi har tolkat uppgift 2 som att vi ska göra om den befintliga koden för att kunna ta in en sträng av en ekvation och programmet ska kunna lösa denna. Den ekvationen som uppgiften specifikt ska lösa är `9*c=5*(f-32)`. Detta innebär att programmet måste kunna hantera multiplikation samt parenteser, samt okända variabler. Vi behöver använda oss av programmet i uppgift 1 för att lösa uppgiften.

Vad var svårt eller lätt med uppgiften?

Det svåra med uppgiften var att förstå koden helt och hållet för att veta vilka partier man skulle ändra för att lösa vissa problem. Under hela utvecklingsprocessen använde vi oss konstant av terminalens felmeddelanden för att hitta “nästa” problem efter att vi löst det föregående. Det var också lite problematiskt att implementera uppgift 1 i uppgift 2 till en början, då det var mycket som inte riktigt gick som vi ville. Vi behövde ändra vissa grejer i uppgift 1 trots att vi var nöjda med lösningen för den specifikt, för att få uppgift 2 att fungera. Att implementera en egen klass för division och subtraktion underlättade en hel del då vi slapp göra massor av problemlösning för att få det att fungera med endast två klasser.

Vi hade också ett problem med att när vi satte den okända variabeln i högerledet till en nämnare i en division, så blev det samma resultat som om den skulle vara täljare. Dock löste sig detta mirakulöst. Vi har ingen aning om hur vi löste det, dock hade vi hållt på och småpillat med koden mellan testerna.

Ett annat problem vi har är att om man ger ett värde till en okänd variabel i vänsterled som inte är i en division, och det är en division i högerled, då måste man ta gånger i divisionen för att lösa det. Detta fungerar inte med vårat program. Om man byter plats i matchningsdelen i constraint-parser2.rb med multiplier och division så fungerar det dock.

Att komma på tester för denna uppgift var inte heller jättelätt, det var svårt att komma på saker för att testa varje enskild funktion för sig. P.g.a. tidsbrist hann vi testa det som uppgiften specifikt bad om, och mycket funktionalitet är testat under utvecklingsprocessen, till exempel addition, subtraktion och division, men vi hann inte skriva många enhetstester, utan bara basic tester som testar basic funktionalitet. Vi har dock med ett testfall som inte fungerar för att visa att ovanstående problem finns.