Individuell uppgift 1

Rapport kodning

FizzBuzz

En loop som går från 1 till 100. Är talet delbart med både 3 och 5 så skrivs FizzBuzz ut, annars kontrolleras om det är delbart med 3 för att skriva ut Fizz, eller med 5 för att skriva ut Buzz. Är inget påstående sant så skrivs talet ut.

Dubbletter

Jag hårdkodar en lista med blandade tal, vissa dubbletter, och skickar in denna i en metod som returnerar en LinkedHashSet. En sådan kan ej innehålla dubbletter samt utför ingen sortering av innehållet.

Rövarspråket

En loop som kör strängens längd. If-satsen kräver sin förklaring. Jag raddar upp alla konsonanter, stora och små. Dessa utgör då en sträng och jag kan använda metoden contains(). I contains kommer alla bokstäver i strängen att kontrolleras. Då contains kräver en sträng som in-parameter lägger jag till

```
+ ””
```

En char plus en sträng blir en sträng! Trots att strängen jag plussar på inte har något innehåll!

Om konsonant hittas så läggs denna till i den nya strängen translated, samt ett ’o’ och samma konsonant igen. Dock gör jag den andra konsonanten till lowercase, vare sig det var stor bokstav eller inte. Den första får ha samma case som källan.

Om ingen konsonant hittas så läggs vokal, eller specialtecken, till i translated.

Min första version jobbade med char-array. If-satsen såg ut som kriget, eftersom den kollade om

```
”c == a || c == A || c == b || c == B…..”
```

Därför gjordes förändringen till ”contains” som ger kortare kod. Dock är den lite mer svårläst…

Rövarspråket 2

Egentligen samma som förra, men skillnaden är att när konsonant hittas så läggs denna till i den nya strängen, och de två nästkommande bokstäverna hoppas över, eftersom de består av ”o + konsonant”. Hittas något annat än en konsonant så läggs denna till i nya strängen och nästa tecken kontrolleras.

Riskorn

Detta var det enda av alla uppgifter som jag inte varit med om tidigare. Utan att googla kom jag dock ganska snabbt fram till att första rutan innehåller 2^0 riskorn, andra 2^1, tredje 2^2 osv. Jag skapade en variable ”rice” som representerar antalet riskorn som ligger i aktuell ruta, 2^i, samt en variabel ”sum” som håller räkningen på hur många riskorn det är totalt. En if-sats kollar om inmatat värde är mindre än summan som loopen kommit upp i. När den är det så returneras ”i + 1” vilket är rutan man kommit till.

Jag fick snabbt byta från ints till doubles, då värdena blev mycket högre än vad jag tänkt. Loopen körs så länge i är mindre än 64, dvs hela schackbrädet, men här skulle man ju kunnat sätta ett helt annat villkor, typ så länge ”input < sum” och bryta på så sätt istället. Men som det är nu är jag hyfsat nöjd då koden är lätt att tolka.

Jag har inte gjort JUnit-tester, men ”testat” med olika inputvärden för att få koden korrekt. Självklart har jag provat med det största möjliga antalet riskorn, 18,446,744,073,709,551,615, men jag får det inte att funka! När jag gör utskrifter i consolen så är summan 1.8446744073709552E19. Flera hundra fel! Kanske ”lite” när vi snackar 18 triljoner, men… Jag hade önskat att jag kunde sätta maxvärdet, för att varna användaren om för högt tal skickas in. Jag har lagt 30 min på detta problem men vill inte lägga mer tid på det nu.