

Inlämningsuppgift 3 Balazs Harko VG-del

Datum: 17-09-2020

Kurs: Javautveckling

Klass: Java20B

Student: Balazs Harko

Introduktion

Jag har skrivit ett litet spel som är inspirerat av en övning i boken som handlar om anagram (Uppgift 8.11, sidan 176).

Spelet går ut på att man ska gissa vilket ord som är skrivet på skärmen, då dess bokstäver har blivit omkastade. Man får 10 frågor och olika poäng beroende på hur långa orden är. Sedan kan man om man vill spara sin poäng till en scoreboard, som är sparad som en textfil i projektmappen så att poängen finns kvar även när programmet avslutas.

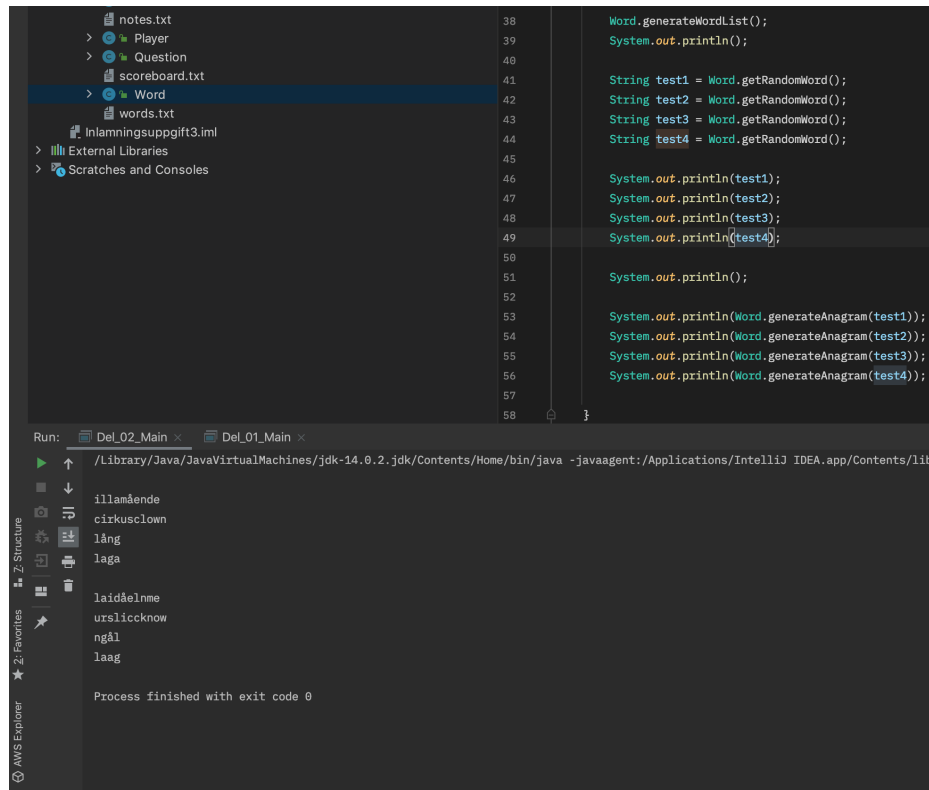
Genomförande

Jag fick börja med att hitta ett sätt att spara en liten lokal ordlista i projektmappen. Jag hittade några ord från ett charader-spel på denna sida: <https://www.godatider.se/familjens-fritid/barnkalas/100-ord-till-charader>

Dem sparade jag i en textfil i projektmappen som jag döpte till *words.txt*, för att sedan läsas in av programmet till en lista.

Jag skapade en klass som jag döpte till *Word* för att behandla allt med ord och även skapandet av anagram. I den sparade jag även alla ord från *words.txt* med hjälp av en metod som läser in filen till en statisk lista. Med hjälp av *Random* klassen kunde jag sedan plocka ut ett slumpvalt ord genom listans *.get()*-metod och ange ett passande slumpvalt index. Sedan skrev jag en metod för att kasta om bokstäverna i ordet (*String*) som den tog emot och returnera ordet med omkastade bokstäver.

(Testar metoderna `Word.getRandomWord()` och `Word.generateAnagram()`)



```
38 Word.generateWordList();
39 System.out.println();
40
41 String test1 = Word.getRandomWord();
42 String test2 = Word.getRandomWord();
43 String test3 = Word.getRandomWord();
44 String test4 = Word.getRandomWord();
45
46 System.out.println(test1);
47 System.out.println(test2);
48 System.out.println(test3);
49 System.out.println(test4);
50
51 System.out.println();
52
53 System.out.println(Word.generateAnagram(test1));
54 System.out.println(Word.generateAnagram(test2));
55 System.out.println(Word.generateAnagram(test3));
56 System.out.println(Word.generateAnagram(test4));
57
58 }
```

Run: Del_02_Main x Del_01_Main x

```
/Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib
illamående
cirkusclown
lång
laga
laidåelnme
urslicknow
ngål
laag
```

Process finished with exit code 0

Jag skapade sedan en `Player`-klass där programmet sparar information om användaren för att kunna spara namn, poäng och tidpunkt när spelaren spelade.

Sedan började jag skriva själva spelet. Då skapade jag en `Game`-klass som hanterar all spelrelaterad logik. När man skapar ett `Game`-objekt skapas även ordlistan i `Word`-klassen med `generateWordList`-metoden. Som parameter anges ett `Player`-objekt.

I `Game`-klassen skapade jag en instansmetod som heter `startGame()` som startar spelet. I den visas ett anagram skapat av metoderna i `Word`-klassen (`getRandomWord()` och `generateAnagram()`).

Sedan ber den användaren att mata in en sträng i terminalen. Om strängen matchar ordet som skickades till `generateAnagram`-metoden får användaren poäng som adderas genom en setter-metod i `Player`-klassen.

Jag ville efter det skapa en meny där man kunde välja att börja spela, se scoreboarden och avsluta. Jag skapade en `Menu`-klass för att hantera allt relaterat till menyer. Vad som tog lång tid var att snygga till alla utskrifter.

De flesta utskrifterna i menyn sparade jag som konstanter i `Menu`-klassen.

I `Menu`-klassen sparade jag även en statisk lista fylld med `Player`-objekt som är scoreboarden. Detta var det svåraste att få till. Jag började med att skriva en metod som skapade filen `scoreboard.txt` om den inte existerade i projektmappen. Hur man kollade om en fil existerade hittade jag här:

<https://howtodoinjava.com/java/io/how-to-check-if-file-exists-in-java/>

För att sedan kunna lägga till spelare till scoreboard-Listan, och sedan kunna se den i menyn, sorterat efter poäng behövde jag läsa in *scoreboard.txt* och sedan konvertera varje rad till Player-objekt. På så sätt kunde jag jämföra olika spelares poäng med `getScore()`-metoden i Player-klassen. Jag skrev metoden `addToScoreBoard()` för att spara ett Player-objekt som en sträng till *scoreboard.txt*. I Player-klassen skrev jag om `toString()`-metoden så att ett player-objekt skrivs ut som en sträng i ett passande format till *scoreboard.txt*:

Bali Test

6

Sep 16 2020 - 22:29:40

Jag skrev sedan metoden `fillScoreBoardList()` som läser in *scoreboard.txt* rad för rad och anropar metoden `readPlayerFromFile()` med `raden(sträng)` som parameter. Jag skapade `readPlayerFromFile()`-metoden i Player-klassen där strängen som läses in konverteras till ett Player-objekt med en privat konstruktör (Jag använde mig av sidan 218 i boken för att se hur jag hanterar dubbelnamn innan poängen läses in som en integer till Player-objektet). Jag använde `.substring()`-metoden för att få ut datumet som en separat sträng. `readPlayerFromFile()` returnerar sedan ett nytt Player-objekt med all information sparad. Detta händer för varje rad i *scoreboard.txt*.

I Menu-Klassen skrev jag sedan metoden `showMenu()` som visar och navigerar runt i menyn.

Min plan var att egentligen bara visa menyn med metoden men det blev naturligt att använda den för all navigering i menyn. Som hjälp skrev jag metoden `menuSelect()` för att använda och returnera giltiga integers för att navigera. Jag kunde på så sätt hantera felaktiga inmatningar där på ett smidigt sätt.

Jag skrev även en liten metod för att rensa terminalen (`clearTerminal()`). Jag visste inget annat sätt att rensa terminalen på, så den radbryter 80 gånger med en for-loop. Det känns lite lyxigt men också klumpigt.

Jag använde sedan en switch-sats för att kunna välja sida i menyn.

Sedan skrev jag en liten beskrivning om spelet i Main-klassen, som jag tyckte var så fin att jag även lade till den i menyn och när spelet startar:

```
***** GISSA ORDET-SPELET *****
*           Gissa vilket ord som visas i terminalen:           *
*                                                                 *
* - Varje ord har fått bokstäverna omkastade.                  *
* - 10 frågor.                                                  *
* - Ord med 3 eller färre bokstäver ger 1 poäng.              *
* - Ord med minst 4 bokstäver och under 6 bokstäver ger 2 poäng. *
* - Ord med 6 eller fler bokstäver ger 3 poäng.                *
*                                                                 *
*                           LYCKA TILL!                          *
*                           //Bali                             *
*****
Tryck på [Enter] för att gå vidare. (Är du inte längst ner i terminalen får du trycka två gånger)
```

Menyn såg sedan ut så här:

```
*****
*           - MENY -           *
*****
Ange 1, 2, 3 eller 4 för att:

1. Spela!
2. Visa Scoreboard
3. Se beskrivning
4. Avsluta
```

När användaren vill börja spela får man ange sitt namn. Det skickas då som en sträng till en konstruktor som skapar ett Player-objekt som sedan skickas till Game-konstruktor. För att undvika strul bestämde jag att namn inte kan innehålla siffror. Jag skrev metoden nameNoNumber() - som endast returnerar en sträng om den inte innehåller en siffra - för att kontrollera detta.

För att visa scoreboard i terminalen skrev jag metoden showScoreBoard(), som med en foreach-loop skriver ut alla resultat i scoreBoard-listan till terminalen.

Om användaren väljer att avsluta gjorde jag en liten extra prompt:

```
* * * * *
*               - MENY -               *
* * * * *
Är du säker på att du vill avsluta?
Ange 1 eller 2:

1.  Ja
2.  Nej
```

Väljer användaren att avsluta kallas `System.exit()`.

Jag lade in ett litet meddelande om man gick in på "Visa Scoreboard" och `scoreboard.txt` saknades eller var tom. Om filen saknades visas ett litet meddelande som säger att filen har skapats:

```
Ingen fil hittades. Scoreboard-fil skapad (.txt)
* * * * * SCOREBOARD * * * * *
Spelare:          Poäng:          Datum:
* * * * *

Inga poäng sparade ännu...

Tryck på [Enter] för att gå tillbaka till huvudmenyn.
```

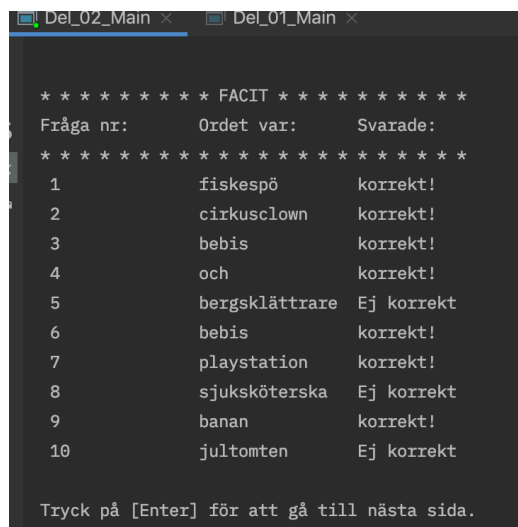
Jag ville sedan hitta ett bra sätt att spara användarens svar på varje fråga i Game-klassen för att sedan kunna visa upp resultatet efter en runda. Jag skapade då en ny klass som jag döpte till `Question` som sparar frågans nummer, originalordet och om användaren svarade rätt eller fel. Efter varje fråga skapas ett nytt `Question`-objekt som sedan sparas i en array med `Question`-objekt.

Sist skrev jag metoden `showAfterGameMenu()` i `Menu`-klassen för att visa resultatet för användaren efter en spelad runda. Den tar emot `Player`-objektet med information om användaren som har spelat och arrayen med `Question`-objekt(resultaten) och skriver ut det i terminalen. Jag skrev sedan en prompt som frågar om användaren vill spara resultatet till scoreboarden.

Jag använde då igen min metod `menuSelect()` för att kontrollera att inmatningen blir korrekt. Jag gjorde om lite i `menuSelect()` så att den tar emot en boolean. Om den är `true` accepterar den enbart 1 och 2. om `false` tar den istället emot 1 - 4. jag sparade dessa booleans som konstanterna `MENU_PROMPT` och `TWO_PROMPT`.

Om man väljer att spara sitt resultat kallas `addToScoreBoard()`- metoden och resultatet skrivs till `scoreboard.txt`.

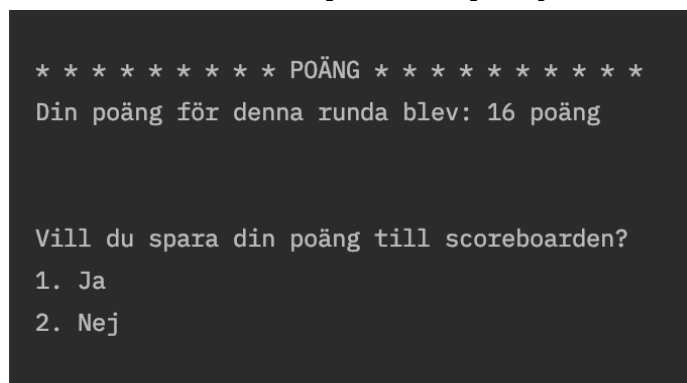
När rundan är slut visas resultatet upp i terminalen:



```
***** FACIT *****
Fråga nr:      Ordet var:      Svarade:
*****
1            fiskespö         korrekt!
2            cirkusclown      korrekt!
3            bebis            korrekt!
4            och              korrekt!
5            bergsklättrare   Ej korrekt
6            bebis            korrekt!
7            playstation      korrekt!
8            sjuksköterska    Ej korrekt
9            banan            korrekt!
10           jultomten        Ej korrekt

Tryck på [Enter] för att gå till nästa sida.
```

Om användaren vill spara sin poäng till scoreboarden:



```
***** POÄNG *****
Din poäng för denna runda blev: 16 poäng

Vill du spara din poäng till scoreboarden?
1. Ja
2. Nej
```

Om man vill addera ord till ordlistan är det bara att fylla på `words.txt` med fler ord.

Vill man nollställa scoreboarden är det bara att radera `scoreboard.txt`-filen i projektmappen så skapas en ny, tom fil nästa gång man spelar.

(Fuska inte genom att skriva i `scoreboard.txt`-filen!)

Utvärdering

Jag är nöjd med resultatet. Spelet verkar funka som det ska och programmet sparar resultaten och läser filerna som det ska.

Vissa saker känns lite klumpiga med navigeringen i terminalen, t.ex. att man måste klicka i terminalen i IntelliJ först, för att sedan kunna mata in ett värde, gör att man ibland måste trycka på [Enter] två gånger när man startar spelet.

Det hade varit häftigt om man kunde utöka programmet med ord inom ett visst ämne genom att använda internet för att fylla ordlistan istället för en `.txt`-fil i projektmappen. Nu kommer orden från en hemsida med tips på ord för charader på barnkalas som jag kopierade till en `.txt`-fil. Utan `words.txt` får man inga ord in i programmet och spelet funkar då inte.

Jag hade velat hitta ett bättre sätt att spara spelarnas resultat i en fil. Med `scoreboard.txt` är det jättelätt att fuska. Det är bara att fylla i vilka värden man vill och skryta för sina kompisar.

Det hade varit snyggare med en `clearTerminal()`-metod som rensar terminalen på riktigt istället för hur jag löste det nu med radrytningar.

Jag hade velat ha ett sätt att kunna ha siffror i namnen och sedan kunna läsa in poängen, men det blev för mycket att koda om kände jag och tiden blev knapp.

Processen var ganska smärtfri ändå. Jag fastnade ett tag på scoreboard och hur jag skulle spara resultaten, men efter att ha provat några olika sätt att använda `.substring` kom jag ihåg att det stod i boken om att använda Scanner-klassens `.hasNext()` - metoder. Det hjälpte oerhört mycket och det blev ganska smidigt ändå.

En klasskamrat visade mig lite snabbt hur man använder sig av breakpoints och debuggern i IntelliJ. Det var utan tvekan den största hjälpen när jag fastnade någonstans i logiken.

Jag försökte att fånga alla fel jag kunde hitta, men det finns nog en hel del kvar!

/Bali

Slutsatser

Jag tror nog att jag tog mig lite vatten över huvudet med vad jag ville bygga först, men lyckades stanna upp tänka över programmet innan jag gick in för djupt och allt blev trassel! Jag hade en plan på ett alternativ med flera spelare och även en tidskomponent där man även får poäng för hur snabbt man svarar på frågorna, men jag är glad att jag hoppade över dem nu.

Uppgiften fick mig att få bättre överblick på hur jag ska skapa klasser och metoder av olika slag för att prata med varandra, men det var lätt att jag trasslade in mig i massor av kod som förmodligen hade kunnat vara enklare.

Jag lärde mig mycket och hade väldigt kul!

/Bali