Assignment 27 – A Game for Two Part 2

Doel

Het doel van deze opdracht is het oefenen met classes en objecten, waarbij de focus ligt op het gebruik van een class in een andere class.

Waar we in de Assignment 26 uitsluitend gebruik maakten van de default constructors, gaan we in deze opdracht expliciet gebruik maken van parameterized constructors of te wel custom constructors.

Overzicht

Net als Assignement 26, heeft deze opdracht ook 3 classes. Je kan ze kopieren en aanpassen of opnieuw schrijven in het kader van "oefening baart kunst".

- Class Assignment27
- Class Game
- Class Player

Opmerking: Alleen de methode **printScoreBoard** heeft **System.out.println** methode(s) om informatie te printen naar console.

1. Voorbereiding

a. Maak een package aan met de naam **assignment27** (let op packages beginnen altijd met een kleine letter)

b. Maak in de package een nieuwe class met de naam **Assignment27** en maak de **main** methode aan. De verdere implementatie van de **main** methode wordt verder op in de opdracht besproken.

2. Class Player

We beginnen met het maken van de class **Player**, omdat deze gebruikt gaat worden door de class **Assignment27** en class **Game**.

a. Maak in de package **assignment27** de nieuwe class met de naam **Player**.

b. De class **Player** heeft twee eigenschappen (fields), resp. **name** en **score**. Bepaal zelf meest voor de hand liggend data type. De fields dienen beide **private** te zijn.

c. Maak voor beide fields de bijbehorende getters en setters.

d. De class heeft een custom constructor met 1 parameter, waarmee de **name** wordt gezet bij het instantiëren van een nieuw object. Het data type van deze parameter dient hetzelfde te zijn als in 2b.

3. Class Game

De class **Game** gebruikt de class Player.

a. Maak in de package **assignment27** de nieuwe class met de naam **Game**.

b. De class **Game** heeft drie eigenschappen (fields), resp. **title**, **player1** en **player2**. Bepaal zelf meest voor de hand liggend data type voor field **title**. De fields player1 en player2 dienen beide van het data type Player te zijn, welke we eerder hebben aangemaakt.

c. Alle fields dienen beide private te zijn.

d. Maak voor alle fields de bijbehorende getters en setters.

e. De class heeft een custom constructor met 3 parameter
   - De eerste waarmee de **title** wordt gezet bij het instantiëren van een nieuw object. Het data type van deze parameter dient hetzelfde te zijn als in 2b.
   - De tweede waarmee de eerste player wordt gezet bij het instantiëren van een nieuw object. Het data type dient **Player** te zijn.
   - En de derde waarmee de tweede player wordt gezet bij het instantiëren van een nieuw object. Het data type dient **Player** te zijn.

f. De class Game heeft (naast de getters en setters) een methode getPlayerNameHighestScore (hieronder verder uitgelegd)

g. De class Game heeft een methode **printScoreBoard** (hieronder verder uitgelegd)

Method getPlayerNameHighestScore

De methode **getPlayerNameHighestScore** heeft in zijn naam “get” maar is geen getter omdat deze niet een overeenkomstig field heeft. Maar moet bepalen welke van de twee spelers de hoogste score heeft en wordt uitsluitend gebruikt door de class zelf (door de methode **printScoreBoard**).

a. Pas de enige en juiste access modifier toe op deze methode.
b. De methode geeft de name van de speler met de hoogste score terug.
c. En heeft geen parameters.
d. Maakt gebruik van de fields van de class.
e. Wanneer de score van de beide spelers gelijk is, dan wordt de String “equal score” teruggegeven.

Method printScoreBoard

De methode **printScoreBoard** print het scorebord naar console.

a. De methode geeft niets terug en heeft geen parameters.
b. De output van de methode is hieronder afgebeeld, de implementatie ervan moet jezelf bedenken. Let op maak gebruik van de fields van de class.

```
### Scoreboard of the game Schaken ###
Player one Thomas has a score of 33
Player two Yvette has a score of 50
Player Yvette has the highest score!
```

4. Class Assignment27

De **main** methode hebben we al aangemaakt in de Voorbereiding (1b). Nu gaan we de implementatie doen van de **main** methode.

a. Instantieer van de class **Player**, een nieuw object met de naam player1 en gebruik de constructor om direct bij instantiëren de naam “Thomas” en zijn score 33 te zetten.

b. Instantieer van de class Player, een nieuw object met de naam player2 en gebruik de constructor om direct bij instantiëren de naam “Yvette” en haar score 50 te zetten.

c. Instantieer van de class **Game**, een nieuw object met de naam **chess** (schaken) en gebruik de constructor om direct bij instantiëren de title “Schaken” en de twee players te zetten.

d. Print het score bord uit van het object **chess**, of anders gezegd print het score bord uit van het spel **chess**.

Extra:

- Instantieer een nieuw spel en geeft het spel een titel
- Instantieer twee nieuwe spelers met nieuwe namen en scores
- Print het score bord
- Pas de naam en de score aan van 1 van de speler m.b.v. de bijbehorende setters
- Print opnieuw het score bord
- Merk op dat je dus meerdere spellen kunt maken met verschillende spelers, maar wel gebruik maakt van dezelfde classes, dat is 1 van de krachtige eigenschappen van OOP!

5. Opmerking

Na het maken van de opdracht zul je wellicht opmerken dat bepaalde getters niet gebruikt worden. Deze mogen verwijderd worden, maar kunnen handig blijken wanneer een bepaald field opgevraagd moet worden.

In deze implementatie van deze opdracht worden ze niet gebruikt, maar stel dat iemand anders gebruik wil maken van jou classes en een bepaald field wil opvragen.