Aix*Marseille université
Socialement engagée
Année 2021-22

Domaine Sciences et Technologies
Portail René Descartes

Programmation 1 : TD 10
Code UE : SPO2U07L

TD 10 : examen blanc

1 Question de cours

Répondez sans explications aux questions suivantes.

Question 1 : Soit `s1` et `s2` deux variables de type `String`. Comment peut-on tester que les deux chaînes de caractères contiennent les mêmes caractères dans le même ordre ?

Question 2 : Quelle instruction est rendue impossible (en dehors du code des constructeurs ou de l'initialisation par défaut) par l'utilisation du mot réservé `final` dans la déclaration d'un attribut `myAttribute` ?

Question 3 : Écrire une expression pour le cosinus de $\pi/4$ en Java.

2 Compter les points au tennis

Nous nous intéressons au comptage des points dans un sport, avec le cas du tennis comme application.

3 Classe `Player`

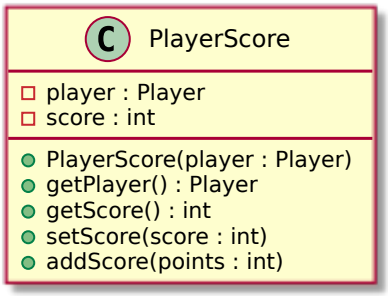
Question 4 : Compléter la classe `Player`, représentant les joueurs. Chaque joueur possède un nom de famille (*lastName*) et un prénom (*firstName*). La méthode `getName()` permet de récupérer le nom complet (prénom et nom de famille séparé par un espace), la méthode `toString()` retourne simplement le nom de famille.

```
public class Player {
  // Add attributes
  
  public Player(String firstName, String lastName) {
    
  }
```

```
  public String getName() { // exemple : "Ashleigh Barty"
    
  }
  public String toString() { // exemple : "Barty"
    
  }
  public boolean equals(Object o) {
    if (!o instanceof Player) {
      return false;
    }
    Player other = (Player) o;
    
  }
}
```

4 Classe `PlayerScore`

Pour pouvoir attribuer un score à un joueur, nous allons utiliser la classe `PlayerScore` dont les objets contiennent le joueur et son score.



Une nouvelle instance de `PlayerScore` contient initialement un score de 0. La méthode `addScore` permet d'ajouter des points au score du joueur. Par exemple, si Alice avait 2 points, et qu'on appelle `aliceScore.addScore(4)`, après l'instruction Alice aura 6 points.

Question 5 : Écrire la méthode `addScore`.

5 Classe `ScoreRecorder`

Afin de comptabiliser les points marqués par tous les joueurs, nous introduisons une classe dont les objets mémorisent les scores de tous les joueurs. On peut comprendre cela comme le tableau d'affichage des scores. Cette classe possède des méthodes permettant d'ajouter un ou plusieurs points à un joueur de notre choix (`givePoint`, `givePoints`), de récupérer le score d'un joueur (`getScore`) ou encore d'obtenir la somme des

scores de tous les joueurs (`getTotalPoints`). La méthode `getPlayers` permet d'obtenir la liste des joueurs ayant un score (chaque joueur ne devant apparaître qu'une fois).

L'attribut `scores` est une liste d'objets instances de la classe `PlayerScore` pour stocker les scores. Elle contiendra donc un objet pour chaque joueur.

La méthode privée `find` appliquée au joueur Bob cherche l'instance de `PlayerScore` contenant Bob et la retourne. S'il n'y a pas de telle instance, `find` doit instancier un objet `PlayerScore` pour Bob, avec un score de `0`, l'ajouter dans la liste `scores` et le retourner.

Les méthodes `getScore`, `givePoint` et `givePoints` doivent utiliser la méthode `find`. Si le joueur avait déjà des points, les nouveaux points lui sont ajoutés en plus, par exemple si un joueur qui avait 3 points en reçoit 5 de plus, son nouveau score est de 8.

Pour l'instant, la classe n'a pas de constructeur.

Question 6 : Compléter les méthodes de la classe `ScoreRecorder`.

```
public class ScoreRecorder {
  private final List<PlayerScore> scores = new ArrayList<>();

  public int getScore(Player player) {
    
  }
  public void givePoint(Player player) {
    
  }
  public void givePoints(Player player, int points) {
    
  }
  private PlayerScore find(Player player) {
    for (                                    ) {
      if (                                    ) {
        return 
      }
    }
    
  }
  public int getTotalPoints() {
```

```
    }
    public List<Player> getPlayers() {
        
    }
}
```

6 La classe `StandardGame`

Nous nous intéressons maintenant au décompte des points dans un jeu au tennis[1]. Un jeu au tennis est constitué de points. Pendant un jeu, un des joueurs est appelé *serveur* (en anglais *server*), l'autre joueur est le *receveur* (en anglais *receiver*).

La classe `StandardGame` permet de représenter un jeu au tennis. Nous introduirons aussi plus tard une interface `TennisGame` dont `StandardGame` sera une implémentation.

Voici l'implémentation provisoire de `StandardGame`. La méthode `scoreNextPoint` permet d'attribuer un point à un joueur. La méthode `call`, dont l'implémentation sera l'objet d'une question, permet de retourner un texte représentant le score tel qu'annoncé par l'arbitre.

```
public class StandardGame implements TennisGame {
  private final Player server;
  private final Player receiver;
  private final ScoreRecorder scores = new ScoreRecorder();
  private int nbPoints = 0;

  public StandardGame(Player server, Player receiver) {
    this.server = server;
    this.receiver = receiver;
```

1. Le tennis est un sport de raquette se jouant entre deux adversaires. Le système de points du tennis est particulièrement compliqué : une partie (en anglais `match`) est divisée en *manche* (en anglais `set`), chaque manche est divisée en jeu (en anglais `game`), chaque jeu est constitué de plusieurs points. Pour gagner la partie, il faut gagner un certain nombre de manches, pour gagner une manche il faut gagner un certain nombre de jeux, et pour gagner un jeu il faut gagner un certain nombre de points.

```
  }

  public void scoreNextPoint(Player player) {
     this.scores.givePoint(player);
     this.nbPoints = this.nbPoints + 1;
  }

  public ScoreRecorder getScores() {
    return this.scores;
  }

  public Player getServer() {
    return this.server;
  }

  public String call() { ... }
}
```

> **Question 7 :** Dessiner le diagramme de classe pour pour chacune des classes `Player`, `StandardGame` et `ScoreRecorder`.

7 Diagramme d'objets

Voici un exemple de programme qui sera réalisable avec les classes que nous allons continuer d'écrire, et un résultat d'exécution possible :

```
public class Main {
  private static final Random gen = new Random();

  public static void main(String[] args) {
    Player ashleigh = new Player("Ashleigh", "Barty");
    Player naomi = new Player("Naomi", "Osaka");
    TennisGame game = new StandardGame(ashleigh, naomi);

    System.out.println("Service " + game.getServer());
    while (!game.getScores().isOver()) {
      Player scorer = gen.nextBoolean() ? ashleigh : naomi;
      System.out.print("Point " + scorer + ": ");
      game.scoreNextPoint(scorer);
      System.out.println(game.call());
    }
  }
}
```

```
Service Barty
Point Barty: 15-0
Point Osaka: 15-A
Point Osaka: 15-30
```

```
Point Barty: 30-A
Point Osaka: 30-40
Point Barty: 40-A
Point Osaka: Avantage Osaka
Point Osaka: Jeu Osaka
```

> **Question 8 :** Donnez l'état des objets créés (valeurs des attributs des objets) après l'exécution des trois premières instructions de la méthode `main`.

On représente les collections par une liste des références stockées dans la collection, les tables d'associations par la liste des paires de références correspondants aux associations, et les chaînes de caractères par la chaîne entre guillemets.

8 Conditions de victoire

De nombreux sports ont une condition de fin de partie basée soit sur le temps écoulé (comme au football), soit sur le score atteint (comme à la pétanque, au tennis, au volley-ball). Dans ce dernier cas, on souhaite pouvoir exprimer sous quelle condition de score une partie est terminée.

À titre d'exemple, une équipe gagne à la pétanque si elle a marqué au moins 13 points. On pourrait donc écrire la classe suivante permettant de déterminer si un joueur gagne une partie de pétanque :

```
public class PetanqueDecider implements WinDecider {
  public boolean isWinner(Player player, ScoreRecorder scores) {
    return scores.getScore(player) >= 13;
  }
}
```

> **Question 9 :** Afin de pouvoir écrire toute sorte de conditions de victoire, écrire l'interface `WinDecider` dont `PetanqueDecider` est une implémentation.

9 Victoire par avantage

Au tennis, un joueur remporte un jeu standard lorsqu'il a au moins 4 points et au moins 2 points de plus que son adversaire.

Ainsi, si le score est de 4 à 2 ou de 5 à 3, le serveur remporte le jeu, mais si le score est de 3 à 1 ou de 5 à 4, le jeu continue. Un jeu peut durer arbitrairement longtemps.

> **Question 10 :** Réaliser une implémentation `AdvantageDecider` de `WinDecider`. Pour cette implémentation, dont le constructeur aura deux paramètres entiers `minPoints` et `delta`, un joueur est vainqueur s'il possède au moins `minPoints` points, et au moins `delta` points de plus que n'importe lequel de ses adversaires.

Ainsi, pour un jeu standard au tennis, la condition de victoire pourra être créée par l'instruction `new AdvantageDecider(4,2)`.

10 Scores et détermination du vainqueur

Nous souhaitons maintenant intégrer à la classe `ScoreRecorder` la détermination du vainqueur de la partie. Pour cela, nous lui ajoutons un attribut `decider` de type `WinDecider`, et un constructeur pour l'initialiser.

```
public class ScoreRecorder{
  private final WinDecider decider;

  public ScoreRecorder(WinDecider decider) {
    this.decider = decider;
  }
  /* ... */
}
```

Question 11 : Écrire 3 nouvelles méthodes publiques dans `ScoreRecorder` :
— `isWinner(Player player)` décide si un joueur donné est vainqueur,
— `getWinner()` retourne un joueur vainqueur s'il y en a au moins un, et `null` sinon,
— `isOver()` décide si la partie est terminée, c'est-à-dire s'il y a au moins un vainqueur.