esprit Se former autrement HONORIS UNITED UNIVERSITIES	Année universitaire : 2023 - 2024 Examen Écrit
Module : Conception Orienté Objet et Programmation Java **Classes :** 3B2→14 - 3IA **Durée :** 1h30 **Nombre de pages :** 6	**Documents autorisés :** Non **Internet autorisé :** Non **Date :** 23/05/2024 **Heure :** 13h00

Remarque : Veuillez répondre sur la feuille de réponse

Exercice I : QCM (4pts)

Une seule réponse est correcte.

1. Quel est le résultat de l'exécution de ce code ?

```
1  public class Main {
2    public static void main(String[] args) {
3      m(1);
4    }
5  }
6
7  public static void m(int x) {
8    try {
9      m2(x);
10     System.out.println(1);
11   } catch (ArithmeticException e) {
12     System.out.println(2);
13   } catch (Exception e) {
14     System.out.println(3);
15   }
16 }
17 public static void m2(int x) throws IOException {
18   System.out.println(4);
19   if (x == 1)
20     throw new IOException();
21   if (x == 0)
22     throw new ArithmeticException();
23   System.out.println(5);
24 }
```

A. 4 3
B. 4 java.io.exception : Impossible d'exécuter le programme Aucun fichier détecté
C. Erreur java.io.exception : Impossible d'exécuter le programme Aucun fichier détecté Processus terminé avec le code de sortie 1
D. 4 1

2. Quel est le résultat de l'exécution de ce code ?

```java
public class Main {
  public static void main(String[] args) {
    ((A)(new B(3))).m();
  }
}
class A {
  private int x;
  public A(int x) {
    this.x = x;
  }
  public void m() {
    System.out.println(x - 1);
  }
}
class B extends A {
  private int y;
  public B(int y) {
    super(y - 1);
    this.y = y;
  }
  public void m() {
    System.out.println(y + 1);
  }
}
```

A. 4
B. Exception: class B cannot be cast to class A
C. 2
D. 3

3. Quel est le résultat de l'exécution de ce code ?

```java
public class Main {
    public static void main(String[] args) {
        Map<String, Integer> map = new HashMap<>();
        map.put("A", 1);
        map.put("B", 2);
        map.put("C", 3);
        for (Map.Entry<String, Integer> entry : map.entrySet()) {
            System.out.print(entry.getKey() + " : " + entry.getValue());
        }
    }
}
```

A. {A :1,B:2,C:3}
B. A:1B:2C :3
C. Le programme lance une exception
D. Aucune des réponses ci-dessus

4. Quel est le résultat de l'exécution de ce code ?

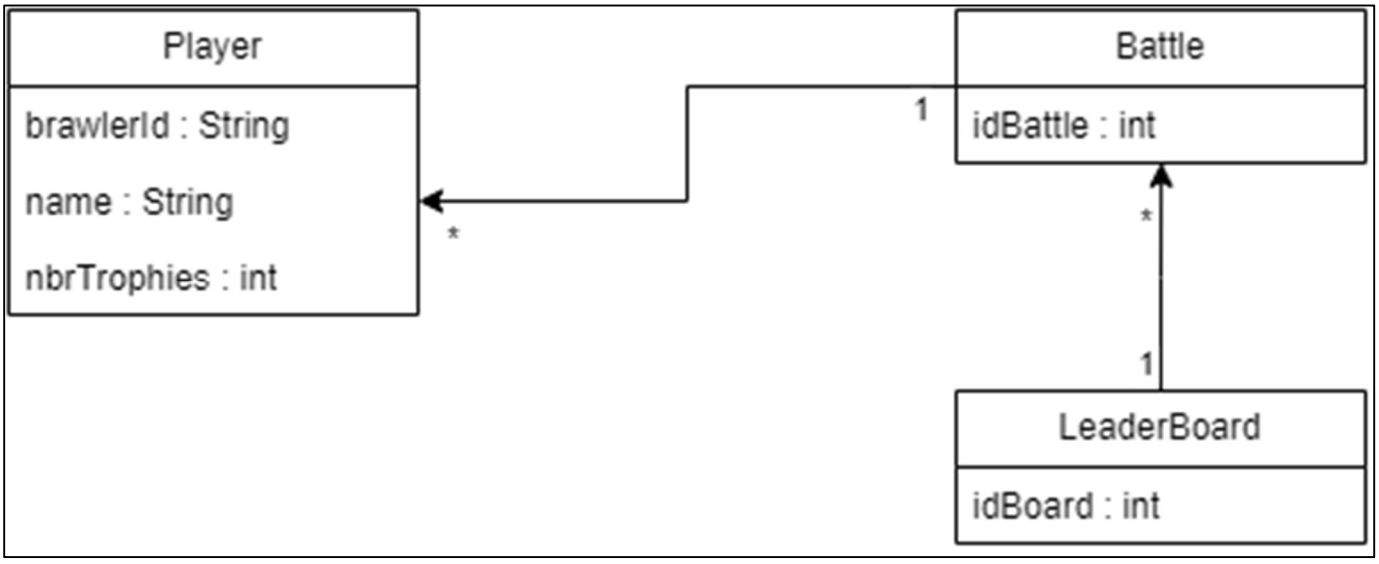
```java
public class Main4 {
    public static void main(String[] args) {
        Set<Integer> numbers = new HashSet<>(Arrays.asList(1, 2, 3, 4, 5));
        List<Integer> evenNumbers = numbers.stream().filter(n -> n % 2 == 0).toList();

        System.out.println(evenNumbers);
    }
}
```

A. [1, 2, 3, 4, 5]
B. [2, 4]
C. [1, 3, 5]
D. Aucune des réponses ci-dessus

Exercice II : Problème (16 pts)

Dans le cadre d'un projet académique, vous avez été assigné à développer un jeu en Java, En vous référant au diagramme de classe fourni et en suivant les instructions données (TODO)



Remarque : les méthodes getters, setters, equals et toString sont déjà implémentées

Classe Battle

```
public class Battle {

// TODO 1 (1 pt) : Déclarez les variables et le constructeur de la classe Battle afin de créer un ensemble de Player sans redondance, nommé players.
……
    public Battle(int idBattle) {
……
  }
// La méthode determineWinner() est déjà implémentée et retourne le joueur gagnant de la bataille.
  public Player determineWinner() {}

//TODO 2 (3 pts) : Implémentez la méthode «addPlayerToBattle» pour ajouter un joueur à une bataille sachant que cette dernière peut contenir au maximum 6 joueurs. Dans le cas où le nombre de joueurs dépasse 6 joueurs, une exception MaxPlayerExceededException sera levée avec le message « Le nombre maximum de joueurs est fixé à 6 ».
   public void addPlayerToBattle(Player player) ……{
……
    }
}
```

Classe Player

```
public class Player {

    private String brawlerId;
    private String name;
    private int nbrTrophies;
```

```
  //TODO 3 (1 pt) : Implémentez la méthode nécessaire pour garantir l'unicité des joueurs
  selon leurs id et leurs noms.
……
……

}
```

Classe LeaderBoard

```
public class LeaderBoard {

 // TODO 4 (1 pt) : Déclarez les variables et une structure, nommé leaderBoard, qui permet
d'associer un score (int) à un joueur et implémentez le constructeur
  public LeaderBoard(int idBoard){
……
 }
//TODO 5 (1,5 pt) : Implémentez la méthode «addPlayerToLeaderBoard» qui permet
d'ajouter le joueur gagnant d'une bataille au leaderBoard et incrémenter le nombre de
trophées. Si le joueur existe, son score sera incrémenté par 1000 sinon le score sera
initialisé à 0.
  public void addPlayerToLeaderBoard(Battle battle){
……
 }
  //TODO 6 (1 pt) : Implémentez la méthode «reportPlayer» qui permet de supprimer un
  joueur du leaderBoard et le sanctionner en décrémentant le nombre de ses trophées par
  1.
  public void reportPlayer(Player p){
```

……

}

//TODO 7 (1,5 pts) : Implémenter la méthode **«sortLeaderBordByTrophies»** qui retourne les joueurs et leurs scores triés dans l'ordre décroissant selon le nombre des trophées.

```
public Map<Player,Integer> sortLeaderBordByTrophies(){
```

……

```
}
```

```
//TODO 8 (2 pts) : Retournez le nombre total des joueurs qui ont plus que 10 trophées (Avec l'API Stream seulement).
public long nbrOfPlayersWithMoreThan10Trophies(){
```

……

```
}
// TODO 9 (2,5 pts) : Ajouter 2 trophées aux 3 premiers joueurs ayant obtenu le score le plus élevé sur le leaderBoard (avec l'API Stream seulement).
  public void rewardTopThreeRankedPlayers(){
```

……

```
}
//TODO 10 (1,5 pts) : Retourner la somme des trophées de tous les joueurs (Avec API Stream seulement).
public int sumOfAllTrophies(){
```

……

```
}
}
```