RN3 - Remise à niveau prog. - Java
LPro - IUT Villetaneuse - Dpt. Info.
Examen - 2020/2021
(Durée: 2 h. 00 min.)

Note: /36

Nom: _____
Prénom: _____

- Le barème est susceptible de changer.
- Aucune correspondance n'est autorisée durant cet examen.
- Vous avez le droit à une feuille (recto seul) de notes dactylographiées.
- Vous répondrez aux questions dans les espaces disposés à cet effet.
- La note tiendra compte de la présentation.
- Toute sortie est définitive.

1. (6 points) **Questions de cours :**
   (durée conseillée : 20 min)

   (a) (2.5 points) Quelles sont les différences entre une *classe concrète*, une *classe abstraite* et une *interface* ?

   Solution:
   - une classe concrète définie des services concrets (avec une implémentation). Il est possible de définir des instances de cette classe
   - une classe abstraite ne peut être instanciées qu'à travers des classes filles concrètes qui proposent une implémentation concrète de tous les service abstraits. Elle peut comporter des service abstraits (signature de méthode associée à aucune implémentation).
   - une interface Définis un contrat purement abstrait. Elle ne peut être instanciée qu'à travers des classes concrètes qui proposent une implémentation concrète de tous les contrats abstraits.

   (b) (2 points) Expliquez chacun des termes de l'expression suivante :
   `public static final int number ;`

   Solution:
   - `public` modificateur de portée (la déclaration est publique), ie la varaible est accessible en lecture et en écriture par n'importe quel programme (de la classe ou d'une autre classe).
   - `static` La déclaration est associée à la classe et non a une instance de la classe. Ici il s'agit d'un variable de classe.
   - `final` La variable est une constance. Une fois initialisé avec une valeur, celle ne peut plus être modifiée
   - `int` La variable est du type atomique `int` qui correspond à un entier signé.
   - `number` C'est le nom de la variable.

   (c) (0.5 points) L'expression de la question (b) suit-elles les conventions en usage chez les développeurs avertis ? Justifiez.

   Solution: Non. Le nom d'un constante est en général donné en majuscules `NUMBER`

   (d) (1 point) Que représente le mot clef `this` ?

   Solution: Dans la déclaration d'une classe `this` représente la référence vers l'instance qui appelle la méthode.

Tot:

[...]

2. (6 points) **QCM**
(durée conseillée : 15 min)
Cochez les cases correspondantes aux affirmations qui vous semblent vraies (toute ambiguïté dans vos marquages des cases seront à votre détriment). Il y a toujours au moins une réponse correcte, mais il peut y en avoir plusieurs. Chaque erreur est pénalisante (oubli d'une bonne réponse, sélection d'un mauvaise réponse).

   (a) Dans l'instruction : `System.out.println("Bonjour");`

      i. (0.5 points) `System` correspond à :
      √ **une classe** ☐ un appel de méthode ☐ une variable d'instance ☐ une variable de classe

      ii. (0.5 points) `out` correspond à :
      ☐ une classe ☐ un appel de méthode ☐ une variable d'instance √ **une variable de classe**

      iii. (0.5 points) `println("Bonjour")` correspond à :
      ☐ une classe √ **un appel de méthode** ☐ une variable d'instance ☐ une variable de classe

   (b) (0.5 points) Une constante doit être :
   ☐ `const` ☐ `abstract` √ `final` ☐ `static` √ **ne peut être modifiée après initialisation** ☐ ne peut être modifiée après compilation ☐ `static`

   (c) (1 point) Dans la classe `Point`, je souhaite déclarer une variable `cptPoint` dont le but est de compter le nombre d'instances de la classe `Point` créés. Je déclare cette variable :
   ☐ `abstract` ☐ `final` √ `static` √ **j'initialise la variable lors de sa déclaration** ☐ j'initialise la variable dans le constructeur

   (d) (1 point) L'instruction `this()` :
   ☐ correpond à un appel au premier constructeur créé dans la classe ☐ peut être utilisé dans toute méthode publique √ **provoque une erreur si la classe ne possède pas de constructeur sans paramètre** ☐ peut être utilisé dans la méthode `main` pour créer un nouvel objet √ **doit être placée avant toute autre instruction dans le constructeur**

   (e) (1 point) Soient les instructions

```
1 int [] tab = new int[5];
2 System.out.println( tab[5] );
```

   ☐ 0 est affiché ☐ `null` est affiché √ **une exception est levée à l'exécution** ☐ une erreur est détectée à la compilation

   (f) (1 point) Soient les instructions

```
1 Integer [] tab = new Integer[8];
2 System.out.println( tab[5] );
```

   √ **0 est affiché** √ **`null` est affiché** ☐ une exception est levée à l'exécution ☐ une erreur est détectée à la compilation

Tot:

3. (10 points) **Les `Joueur` s**
(durée conseillée : 30 min)
On s'intéresse à des compétitions de tennis (ou autres). Un `Joueur` est caractérisé par un nom (`String`), un identifiant (`int`) et un nombre de points (`int`) qui est le total des points acquis. Un match gagné rapporte 1 point, une défaite 0 point.

Implémenter la classe `Joueur` en complétant le code suivant dans les espaces ci-dessous :

```
1  public class Joueur{
2
3      public static int NB_JOUEURS = 0;
4
5      // variables d'instance ou de classe ....  A COMPLETER
6
7      public Joueur(){}
8
9      public Joueur(String nom, int id){ // ... A COMPLETER
10     }
11     public Joueur(Joueur jModele){ // ... A COMPLETER
12     }
13     public void majRes(boolean res){ // ... A COMPLETER
14     }
15     public int getScore(){ // ... A COMPLETER
16     }
17     public String toString(){ // ... A COMPLETER
18     }
19      ...  nbJoueurs() { // ... A COMPLETER
20     }
21 }
```

(a) (1,5 points) Complétez les variables d'instance et/ou variables de classe

Solution:

```
1 private String nom ;
2 private int id ;
3 private int points ;
```

(b) (2,5 points) Complétez le bloc d'instruction du constructeur champ à champs `public Joueur(String nom, int id)`. ATTENTION : assurez vous que le constructeur fasse bien tout ce que l'on attend de lui...

Solution:

```
1 this.nom = nom ;
2 this.id = id ;
3 this.points = 0;
4 Joueur.NB_JOUEURS ++ ;
```

(c) (1,5 points) Complétez le bloc d'instruction du constructeur par copie `public Joueur(Joueur jModele)`

Solution:

```
1 this.nom = jModele.nom ;
2 this.id = jModele.id ;
3 this.points = jModele.points;
```

(d) (1 point) Complétez le bloc d'instruction de la méthode `public void majRes(boolean res)` qui met à jour le nombre de points obtenus par le joueur (+1 si res est vrai c'est-à-dire si le résultat du match est le gain, +0 sinon).

Solution:

```
1 if( res ) {
2     this.points ++
3 }
```

(e) (0,5 points) Complétez le bloc d'instruction de la méthode `public int getScore()` qui retourne le nombre de points acquis par le joueur.

Solution:

```
1 return this.points ;
```

(f) (1 point) Complétez le bloc d'instruction de la méthode `public String toString()` qui propose une chaîne de caractères décrivant l'état courant de l'instance de `Joueur`.

Solution: return(this.nom + " :" + this.id + " :" + this.getScore()) ;

(g) (2 points) Complétez le bloc d'instruction et l'entête complète de la méthode `... nbJoueurs()` qui retourne le nombre de joueurs qui ont été instanciés lors de son appel.

Solution:

```
1 public static int nbJoueurs() {
2     return Joueur.NB_JOUEURS ;
3 }
```

Tot:

4. (14 points) **Le `Tournoi`**
(durée conseillée : 45 min)
ATTENTION : Dans cet exercice il est demandé d'utiliser des tableaux static et pas des `ArrayList` . On implémente maintenant la classe `Tournoi` qui a pour variables d'instance : le nombre de participants inscrits au tournoi et un tableau dont chaque élément est un `Joueur` participant (inscrit). Compléter la classe `Tournoi` :

```
1  class Tournoi{
2      // variables d'instance ou de classe ...  A COMPLETER
3
4      public Tournoi(int nbMaxParticipants){ //  ... A COMPLETER
5      }
6      public Tournoi(Tournoi tModele){ //  ... A COMPLETER
7      }
8      public void saisirMatch(Joueur j1, boolean res1, Joueur j2, boolean res2) {
9      // ... A COMPLETER
10     }
11     public void ajouteJoueurTournoi(Joueur jr) { // ... A COMPLETER
12     }
13     public Joueur getVainqueur() { // ... A COMPLETER
14     }
15     public int nbInscritsTournoi() { // ... A COMPLETER
16     }
17 }
```

Pour information voici un exemple de test (suivi de son exécution) :

```
1  class Test{
2      public static void main(String[] args){
3          Joueur j1 = new Joueur("Tonga",1);
4          Joueur j2 = new Joueur("Nadal",2);
5          Joueur j3 = new Joueur("Monfils",3);
6          Joueur j4 = new Joueur("inconnu",4);
7          System.out.println("nombre de joueurs saisis :"+Joueur.nbJoueurs());
8
9          Tournoi t = new Tournoi(1000);
10         t.ajouteJoueurTournoi(j1);
11         t.ajouteJoueurTournoi(j2);
12         t.ajouteJoueurTournoi(j3);
13         System.out.println("nb de joueurs inscrits au tournoi :"+ t.nbInscritsTournoi());
14
15         t.saisirMatch(j1, true, j2, false);
16         t.saisirMatch(j1, true, j3, false);
17         t.saisirMatch(j2, false, j3, true);
18         Joueur vainqueur = t.getVainqueur();
19         System.out.println("le vainqueur est : "+ vainqueur.toString());
20 }
```

Exécution du `main` défini ci-dessus :

```
1  santini$ java Test
2  nombre de joueurs saisis :4
3  nombre de joueurs inscrits au tournoi :3
4  le vainqueur est : Tonga  nb points gagnes :2
```

ATTENTION : Les méthodes de cette classe ne sont pas indépendantes entre elles et peuvent dépendre des méthodes de la classe `Joueur`. Veillez à bien comprendre le fonctionnement de chaque méthode de la classe pour pouvoir les utiliser dans d'autres méthodes. Le fait de ne pas utiliser une méthode peut faire perdre des points.

On distinguera dans ce modèle le nombre de joueurs participant au `Tournois` (effectivement inscrits), du nombre maximal possible de participants. Le nombre maximal possible de participants n'est pas mémorisé sous forme d'une variable d'instance (ce n'est pas nécessaire).

(a) (1 point) Complétez les variables d'instance et/ou variables de classe

Solution:

```
1 private int nbParticipants ;              // 0,5
2 private Joueur [] participants;      // 0,5
```

(b) (1,5 points) Complétez le bloc d'instruction du constructeur `public Tournoi(int nbMaxParticipants)` avec `nbMaxParticipants` le nombre maximal de participants pouvant s'inscrire au `Tournoi`.

Solution:

```
1 this.nbParticipants = 0;                              // 0,5
2 this.participants = new Joueur [nbMaxParticipants] ;     // 1
```

(c) (4 points) Complétez le bloc d'instruction du constructeur par copie `public Tournoi(Tournoi tModele)`.

Solution:

```
1 this.nbParticipants = tModele.nbParticipants;                          // 0,5
2 this.participants = new Joueur [tModele.participants.length] ;         // 1,5
3 for ( int i = 0 ; i < tModele.nbInscritsTournoi() ; i++ )                // 0,5
4     this.participants[ i ] = new Joueur( tModele.participants[ i ] ) ;   // 1,5
```

(d) (1 point) Complétez le bloc d'instruction de la méthode `public void saisirMatch(Joueur j1, boolean res1, Joueur j2, boolean res2)` où :

- `j1` est le joueur1
- `j2` est le joueur2
- `res2` est le résultat obtenu par le joueur2
- `res1` est le résultat obtenu par le joueur1
- la méthode met à jour le nombre de points de chaque `Joueur` en fonction de leur résultat.

Solution:

```
1 j1.majRes(res1);          // 1
2 j2.majRes(res2);
```

(e) (2,5 points) Complétez le bloc d'instruction de la méthode `public void ajouteJoueurTournoi(Joueur jr)` met à jour le tableau des joueurs participant au tournoi. Si toutes les places d'inscription au `Tournoi` sont déjà prises par

des `Joueurs` la méthode rejette l'inscription (le `Joueur` n'est pas inscrits) et affiche un message d'erreur.

Solution:

```
1 if ( this.nbInscritsTournoi() == this.participants.length ) {    // 1
2     System.out.println("Tournois deja complet!");
3 } else {
4     this.participants[ this.nbInscritsTournoi() ] = jr ;         // 1
5     this.nbParticipants ++ ;                                     // 0.5
6 }
```

(f) (3,5 points) Complétez le bloc d'instruction de la méthode `public Joueur getVainqueur()` qui retourne le vainqueur du tournoi (celui qui a le plus grand nombre de points).

Solution:

```
1 Joueur vainqueur = this.participants[ 0 ];                          // 0.5
2 for ( int i = 1 ; i < this.nbInscritsTournoi() ; i++ ) {                  // 1
3     if ( this.participants[ i ].getScore() > vainqueur.getScore() )   // 1
4         vainqueur = this.participants[ i ];                          // 0.5
5 }
6 return vainqueur;                                                  // 0,5
```

(g) (0,5 points) Complétez le bloc d'instruction de la méthode `public int nbInscritsTournoi()` retourne le nombre d'inscrits au tournoi.

Solution:

```
1 return this.nbParticipants;          // 0,5
```