IFT1176 - Session Été, Intra

Mohamed Lokbani

IFT1176 - INTRA

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Section: _____|

Date : 5 juin 2002

Durée: 2 heures (de 20h30 à 22h30) Local: 1140 du Pavillon André-Aisenstadt (P.A-A).

Directives:

- Toute documentation permise.
- Calculatrice **non** permise.

- Répondre directement sur le questionnaire.

1._____/12

2._____/20

3._____/18

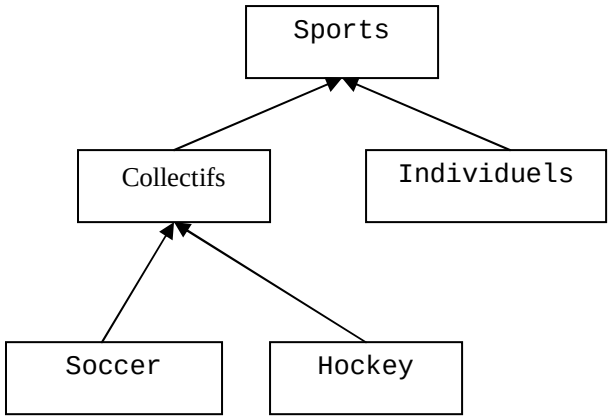
4._____/20

5._____/30

Total:_____/100

Question 1 (12 points)

Supposer la représentation suivante des classes `Sport`, `Individuel`, `Collectif`, `Soccer` et `Hockey` :



`Les classes Individuels et Collectifs dérivent de la classe Sports. Quant aux classes Soccer et Hockey, elles dérivent de la classe « sports collectifs » Collectifs.`

Soit le fragment de code suivant :

```
1   Sports monsport = new Soccer();
2   Individuels karate = new Individuels();
```

Répondre aux questions suivantes :

-**Q1.1**- Est-ce que `monsport` est une instance (instanceof) de la classe `Individuels`?

OUI ☐ **NON** ☐

Pourquoi? (Courte explication)

-**Q1.2**- Est-ce que `monsport` est une instance (instanceof) de la classe `Collectifs`?

OUI ☐ **NON** ☐

Pourquoi? (Courte explication)

-**Q1.3-** Est-ce que `monsport` est une instance (instanceof) de la classe `Soccer`?
OUI ☐ **NON** ☐

Pourquoi? (Courte explication)

-**Q1.4**- Est-ce que `monsport` est une instance (instanceof) de la classe `Hockey`?
OUI ☐ **NON** ☐

Pourquoi? (Courte explication)

-**Q1.5**- Est-ce que `karate` est une instance (instanceof) de la classe `Individuels`?
OUI ☐ **NON** ☐

Pourquoi? (Courte explication)

-**Q1.6**- Est-ce que `karate` est une instance (instanceof) de la classe `Sports`?
OUI ☐ **NON** ☐

Pourquoi? (Courte explication)

-**Q1.7**- Est-ce que `karate` est une instance (instanceof) de la classe `Collectifs`?
OUI ☐ **NON** ☐

Pourquoi? (Courte explication)

Question 2 (20 points)

Soit le fragment de code suivant:

```
1   class Cercle {
2      private double rayon;
3      public Cercle(double rayon) {
4            rayon = rayon;
5      }
6      public double getRayon() {
7            return rayon;
8      }
9      public double calculSurface() {
10           return rayon*rayon*Math.PI;
11     }
12  }
13  class Cylindre extends Cercle {
14     private double longueur;
15     public Cylindre(double rayon, double longueur) {
16           Cercle(rayon);
17           longueur = longueur;
18     }
19     public double calculSurface() {
20           return calculSurface()*longueur;
21     }
22  }
```

-**Q2.1**- Expliquer pourquoi l'appel **Math.PI** (ligne 10) est syntaxiquement correct.

-**Q2.2**- Le précédent fragment de code contient **4 erreurs** de syntaxe ; identifiez-les et corrigez-les. Les numéros de lignes ne font pas partie du programme, elles sont la à titre indicatif. Par ailleurs, des erreurs sur une même ligne compte pour une seule erreur.

Erreur -1-

Ligne :

Erreur :

Courte explication :

Correction :

Erreur -2-

Ligne :

Erreur :

Courte explication :

Correction :

Erreur -3-

Ligne :

Erreur :

Courte explication :

Correction :

Erreur -4-

Ligne :

Erreur :

Courte explication :

Correction :

Question 3 (18 points)

Soit le programme suivant, qui compile correctement :

```
1   public class exo3 {
2       static void test(long uneValeur) {
3               try {
4                   System.out.println("entree bloc try");
5                   if (uneValeur > 100)
6                           throw new Exception (uneValeur + " est trop grande");
7                   System.out.println("sortie du bloc try");
8               } catch (Exception e) {
9                   System.out.println("ERREUR: " + e.getMessage());
10              }
11      }
12      public static void main(String[] args){
13              if (args.length ==1){
14                  System.out.println("L'exemple " + args[0] +
15                                          " va produire le resultat suivant:");
16                  test(Long.parseLong(args[0]));
17              }
18      }
19  }
```

Que vont afficher l'exécution des lignes suivantes (à partir du précédent programme) :

-**Q3.1**- `java exo3 50`
Réponse

-**Q3.2**- `java exo3 199`
Réponse

-**Q3.3**- `java exo3 0.99`
Réponse

Question 4 (20 points)

Soit le fragment de code suivant, utilisant deux HashMap. La première HashMap associe une clé idNombre à un nom, alors que la seconde associe cette même clé à un salaire :

```
import java.util.*;
class exo4 {
    public static void main (String[] args) {
      //Test les données
      HashMap idANom = new HashMap(), idASalaire = new HashMap();
      idANom.put(new Integer(101),"Luc");
      idANom.put(new Integer(102), "Marc");
      idANom.put(new Integer(103), "Carla");
      idASalaire.put(new Integer(101), new Integer(900));
      idASalaire.put(new Integer(102), new Integer(900));
      idASalaire.put(new Integer(103), new Integer(901));
      printTables( idANom, idASalaire);
    }

      /*
      Etc.
      */
}
```

Question

Écrire le code de la méthode **`printTables`**, permettant d'afficher en sortie :

```
{Carla=901, Luc=900, Marc=900}
[Luc=900, Marc=900, Carla=901]
```

- le premier affichage donne en sortie les noms et les salaires triés par nom.
- le second affichage donne en sortie les noms et les salaires triés par salaire.

Vous pouvez supposer que les noms sont uniques, mais que deux personnes peuvent avoir un salaire identique.

Vous pouvez compléter la classe **`exo4`** par d'autres méthodes, si vous jugez cela nécessaire.

Réponse

Question 5 (30 points)

L'interface graphique `Multiplier` permet de calculer le produit de deux nombres entiers `Nombre1` et `Nombre2` et d'afficher le résultat dans la case `Réponse`. Cette interface est dotée aussi des boutons `Multiplier` pour calculer le produit des deux nombres, `Effacer` pour effacer le contenu des cases `Nombre1`, `Nombre2` et `Réponse` ; et finalement le bouton `Quitter` permet de fermer la fenêtre `Multiplier` et quitter ainsi le programme.

Il vous est demandé d'écrire **uniquement** la définition de la méthode `actionPerformed(ActionEvent)` associée à cette interface graphique.

Pour cela, vous avez besoin de tenir compte des trois exceptions suivantes :

-1- IntegerOutOfRangeException

```
public class IntegerOutOfRangeException extends Exception {
      public IntegerOutOfRangeException(String errorMessage) {
            super(errorMessage);
      }
}
```

Cette exception sera levée dans les deux cas suivants :

- Le nombre entré est supérieur à la borne max définie par la constante `Integer.MAX_VALUE` ou inférieure à la borne min définie par la constante `Integer.MIN_VALUE`.
- Le produit `Nombre1xNombre2` est supérieur à `Integer.MAX_VALUE` ou inférieure à `Integer.MIN_VALUE`.

Par exemple si Nombre1= `-2147483649 < Integer.MIN_VALUE (qui vaut -2147483648)`
Une exception sera levée dans ce cas.

-2- FieldIsEmptyException

```
public class FieldIsEmptyException extends Exception {
      public FieldIsEmptyException(String errorMessage) {
            super(errorMessage);
      }
}
```

Cette exception sera levée dans le cas ou vous aviez omis d'entrer une valeur. Exemple vous laissez la case Nombre1 vide, et vous avez appuyé sur le bouton `Multiplier`.

-3- NumberFormatException

Une exception de Java, qui indique que l'application a essayé de convertir une chaîne supposée string mais en réalité, elle ne l'est pas.

Les exceptions seront levées à travers une fenêtre de dialogue du type `showMessageDialog`.

Vous pouvez considérer les déclarations de variables suivantes :

```
JLabel nombre1Label, nombre2Label, reponseLabel;
JTextField nombre1Field, nombre2Field,reponse Field;
JButton multiplierButton, effacerButton, quitterButton;
```

Réponse

```
public void actionPerformed (ActionEvent e) {
```

```
}
```