Université Paris Descartes,
UFR Mathématiques et Informatique
L2 Informatique

Examen POO
(1ere session)

Aucun document autorisé

class abstract

Exercice 1

Répondez par Vrai ou par Faux aux questions suivantes :

Q1 Une interface peut hériter d'une ou de plusieurs interfaces

Q2 Une classe abstraite est une classe qui contient au plus une méthode abstraite

Q3 Les instructions contenues dans un bloc ***finally*** seront exécutées qu'il y ait ou non une exception

Q4. Une classe ne peut implémenter plusieurs interfaces

Q5. Toute classe qui implémente une interface est sous type de cette interface

Q6. Le code ci-dessous affiche le texte "*Hello from Class1*"

```
public class Class1{
  public Class1 (){
    System.out.println("Hello from Class1");
  }
}
public class Class2 extends Class1{}

//Classe principale
public class TestClass2 {
  public static void main(String[] args)
  {
     Class2 c2 = new Class2 ();
  }
}
```

Q7. Une classe qui implémente une interface hérite automatiquement de toutes ses constantes

Q8. Implémenter l'interface *Serializable* revient à implémenter toutes les méthodes qu'elle contient

Exercice 2 Corrigez les morceaux de code ci-dessous

```
public class FormeGeometrique1 {
  double posX, posY;

  double surface() ;
  double perimetre() ;

  public deplacer(double x, double y) {
      posX=x;
      posY=y;
  }
  public afficher() {
    System.out.println("position : ("
                    +posX+","+posY+")");
  }
}
```

Une classe personnalisée, *RectangleException*, a été créée comme sous-classe de *Exception*.

```
class Rectangle{
  int longueur;
  int largeur;

  public Rectangle(int lo, int la) throw RectangleException {
    if (lo < 0 || la < 0)
      throws new RectangleException();
    else{
      longueur = lo;
      largeur = la;
    }
  }
}
```

Exercice 3
Expliquez (en précisant si c'est juste ou faux) le morceau de code ci-dessous

```
//Vecteur d'entiers
Public static void main(String[] args){
  Vector<Integer> v = new Vector<Integer>();
  int valeur = 0;
  int nbVal = Saisie.lireEntier("Donnez le nombre total de valeurs") ;
  for(int i=0 ; i< nbVal; i++){
      valeur = Saisie.lireEntier("Donnez une valeur ?");
    v.add(valeur);
  }
  System.out.println(v) ;
}
```

Exercice 4
Q1 : Que se passe t'il si on exécute le code ci-dessous en ligne de commande, de la manière suivante : "***java Principale 0 5***".

Q2 : Corrigez le problème en proposant deux solutions distinctes.

```
public class Equation {
 private int a,b;

public Equation(int a, int b) {
   this.a=a; this.b=b;
 }

public void afficher() {
   System.out.println(a+" * X = "+b);
 }

public int solution() {
   return b/a;
 }
}
```

0 * X = 5

```
public class Principale{
  public static void main(String args[]) {
    int a=Integer.valueOf(args[0]).intValue();
    int b=Integer.valueOf(args[1]).intValue();
    Equation equation = new Equation(a,b);
    equation.afficher();
    System.out.println("résultat : X = "+ equation.solution());
  }
}
```

exception

Exercice 5 Gestion d'une collection de meubles.

Complétez ou corrigez le squelette de code ci-dessous

```
public class Meuble {
    int annee;

    public Meuble(...){
      //TO DO
    }

    //Affichage des attributs
    public void afficher(){ //TO DO}

   //Retoune la surface du meuble
      //TO DO
    public  double surface()....
}
```

```
/*Une table est un meuble particulier*/
class  Table {
    String couleur;

   public Table(....){
      //TO DO
    }

   //Affichage des attributs
    public void afficher(){
      //TO DO
    }

  //retourne la surface de la table
        //TO DO
   public  double surface()...
}
```

Université Paris Descartes,
UFR Mathématiques et Informatique
L2 Informatique

```
/*Une table ronde est une table particulière
 * qui possède un rayon */
class TableRonde {
   private double rayon;

   public TableRonde(...){
     //TO DO
   }

   //Affichage des attributs
   public void afficher(){
      //TO DO
   }

  //retourne la surface de la table
      // TO DO
   public  double surface()...
}
```

```
//On souhaite gérer une collection de meubles (tables, table rondes, ...). Expliquez en un
mot, le principe de la POO qui est appliqué ici

public class CollectionMeuble {
   private List<Meuble> meubles;

   public CollectionMeuble(){
      //TO DO;
   }

    //Afficher la surface totale de tous les meubles
   //Attention!!! Il faut utiliser l'interface Iterator pour cette question

    public void afficherSurfaceTotale(){
      //TO DO
   }
}
```