Programmation orientée objets

Documents et appareils électroniques interdits. 2 heures.

Exercice 1 (Questions) (8 points)

Question 1 : Quel est l'intérêt d'une machine virtuelle (dans le contexte de Java) ?

Question 2 : Quelle est la différence entre interface et implémentation ? Pourquoi faire cette distinction ?

Question 3 : Qu'est-ce qu'un « package » ? À quoi servent-ils ?

Question 4 : Lors de la redéfinition d'une méthode, comment appeler la méthode de la super-classe ?

Question 5 : Comment une sous-classe peut-elle appeler le constructeur de sa super-classe ?

Question 6 : Quelle est la différence entre une classe abstraite et une interface ?

Question 7 : Qu'est-ce que l'héritage ? Quelles alternatives peut on utiliser ?

Question 8 : Qu'est-ce que l'encapsulation ? En quoi est-ce utile vis-à-vis des objectifs de l'objet ?

Question 9 : Qu'est-ce qu'une instance ?

Question 10 : Pourquoi utiliser un `enum` plutôt que des constantes ?

Question 11 : En Java, quelle est la différence entre l'opérateur == et la méthode `equals` ?

Question 12 : Qu'est-ce qu'une classe anonyme ? Quel en est l'intérêt ?

Question 13 : Quels sont les intérêts d'utiliser des accesseurs ?

Question 14 : Quelle est la différence entre `throw` et `throws` en Java ?

Question 15 : Quelle est la différence entre une `ArrayList` et une `LinkedList` en Java ? Donner les avantages et incovénients de chacune.

Question 16 : En Java, qu'est-ce que le « boxing » et le « unboxing » ?

Exercice 2 Soit le programme Java

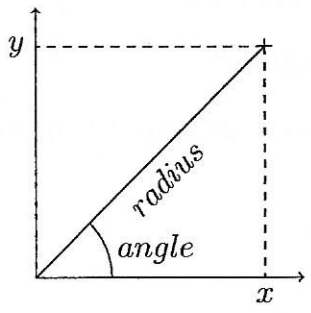
```
class Base {
  public String className = "Base";
}
class Derived extends Base {
  private String className = "Derived";
}
public class Test {
  public static void main(String[] args) {
    System.out.println(new Derived().className);
  }
}
```

Question 1 : (1 pt) Il ne compile pas. Pourquoi ? Comment faire pour qu'il affiche "Derived" ?

```
class Base {
  private String className = "Base";
  public String name() { return this.className; }
}
public class Test2 {
  public static void main(String[] args) {
    System.out.println(new Derived().name());
  }
}
```

Question 2 : (1 pt) Qu'affiche cette version et pourquoi (la classe Derived est inchangée) ?

Exercice 3 (Un point c'est tout !) On se propose ici de mettre en œuvre une classe représentant un point géométrique, dont voici une représentation UML partielle :



Point
name : String {readOnly} x : Double y : Double /angle : Double /radius : Double
move(deltaX : Double, deltaY : Double) : void scale(ratio : Double) : void rotate(angle : Double) : void

x et y représentent les cordonnées carthésiennes, et angle et radius les coordonnées polaires. La méthode rotate change l'angle, et scale multiplie x et y par le ratio donné. move déplace le point (en relatif).

Question 1 (Implémentation) : (2 pt) Donnez le code Java correspondant, en respectant les conventions du langage (attention, le diagramme n'est pas au niveau d'abstraction de l'implémentation, il faudra donc ajouter les éléments nécessaires).

Question 2 (Construction) : (1 pt) On veut pouvoir créer un point en spécifiant soit ses coordonnées carthésiennes, soit ses coordonnées polaires. Comment faire ? Donner le code correspondant.

Question 3 (Égalité) : (2 pt) On veut que deux points soient considérés égaux s'ils sont au même endroit. Comment faire ? Donner le code correspondant.

Question 4 (De la couleur) : (2 pt) On souhaite étendre cette classe en ajoutant de la couleur. Une couleur est représentée par trois valeurs entières (rouge, vert et bleu). Donner le diagramme UML et le code correspondants (la classe Point ne change pas).

Question 5 (Forme) : (1 pt) Une forme est un ensemble de point. Donnez le diagramme UML correspondant. Expliquez la mise en œuvre (pas de code nécessaire).

Question 6 : (2 pt) Comment gérer l'égaliter des points en prennant en compte les points colorés ?