Examen de programmation par objets

Janvier 2000

Licence informatique

Responsable J.Ferber

Tous documents manuels (cours, polys) autorisés. Livres non autorisés

Durée : 2 h

Exercice 1: Héritage et instanciation (9 pts)

Soient les classes suivantes:

```
class Personne {
   public void parler() {
      System.out.println("hum!");
   }
}

class Adulte extends Personne{
   public void parler() {
      System.out.println("Bonjour tout le monde!");
   }
}

class AdulteDistinguee extends Adulte {
   public void parler() {
      System.out.println("Mes chers amis, bonjour!");
   }
}

class Jeune extends Personne {
}

class Ado extends Jeune {
   public void parler() {
      System.out.println("salut les mecs!");
   }
}

class Enfant extends Jeune {
}

class Bebe extends Enfant {
   public void parler() {
      System.out.println("J'chui pas un bebe!");
   }
}

class BebeCadum extends Bebe {
   public void parler() {
      System.out.println("Agheu, agheu!");
   }
}
```

Question 1:

Donnez ce qui est affiché par le programme suivant:

```
main() {
   Personne P1 = new Personne();
   Personne P2 = new AdulteDistinguee();
   Adulte P3 = new AdulteDistingue();
   Personne P4 = new Bebe();
   Jeune P5 = new Ado();
   Enfant P6 = new BebeCadum();
   BebeCadum P7 = new BebeCadum();
   Enfant P8 = new Bebe();

   P1.parler();
   P2.parler();
   P3.parler();
   P4.parler();
   P5.parler();
   P6.parler();
   P7.parler();
   P8.parler();
}
```

Question 2:

On effectue les affectations suivantes (ces affectations sont indépendantes les unes des autres, elles ne sont pas exécutées en séquence!). Pour chaque affectation, dites si elle fonctionne sans erreur, si elle provoque une erreur à la compilation ou si elle provoque une erreur à l'exécution. Si il y a une erreur expliquez (rapidement) pourquoi.

1. `P1 = P2;`
2. `P4 = P1;`
3. `P3 = P4;`
4. `P3 = P1;`
5. `P4 = P5;`
6. `p7 = p6;`
7. `p7 = (BebeCadum) p4;`
8. `p6 = (Bebe) p4;`
9. `p3 = (AdulteDistingue) p2;`
10. `p8 = (Bebe) p5;`

Question 3:

On crée un tableau d'enfants de la manière suivante:

```
Jeune[] e = new Jeune[4];
```

a) Que faut il faire pour que les affectations suivantes soient toutes valides (à la compilation et à l'exécution).

Note: on reprendra les variables dans l'état de la question 1):

```
e[0] =  p4;
e[1] =  p5;
e[2] =  p6;
e[3] =  p7;
e[4] =  p8;
```

b) Que produit l'exécution de l'instruction suivante:

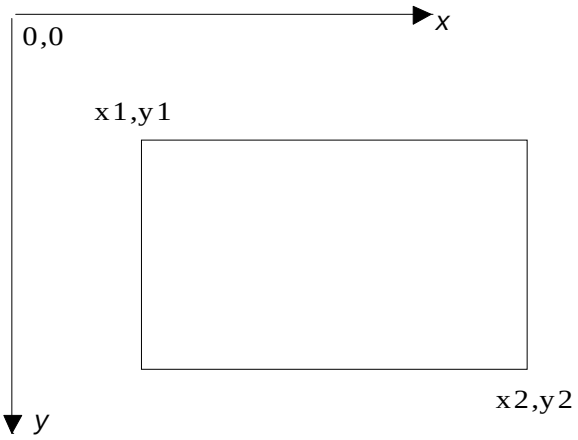
```
for(int i=0;i<e.length;i++)
   e[i].parler();
```

c) Que faut il faire pour que les affectations suivantes soient toutes valides (à la compilation et à l'exécution):

```
p5 = e[0];
p4 = e[1];
p7 = e[2];
p8 = e[3];
p6 = e[4];
```

Exercice 2 : Héritage et envoi de message (11 pts)

On désire réaliser un logiciel de dessin en Java en utilisant la technique objet sans utiliser le package java.awt ou swing de Java. Un dessin est défini par les coordonnées de deux de ses points, (x1, y1, x2, y2) et sa méthode **`dessineToi()`**. La figure suivante montre un dessin avec ses coordonnées (On ne tiendra pas compte des problèmes liés au Graphics. On suppose qu'il n'y a plus de Graphics...).



On suppose que les classes suivantes sont déjà définies (en Java):

```
import java.util.*;

abstract class Dessin extends Object {
      int x1, y1, x2, y2;

      abstract void dessineToi();   // doit etre redefinie

      Dessin(int _x1, int _y1, int _x2, int _y2) {
         x1 = _x1; x2 = _x2;
         y1 = _y1; y2 = _y2;
      }
}
```

```
class DessinComplexe extends Dessin {
      Vector dessins = new Vector();
      void ajoute(Dessin d) {
         dessins.addElement(d);     // ajoute un dessin
      }

      void dessineToi() {
         //à définir
      }
}

class DessinElementaire extends Dessin {
}

class Rectangle extends DessinElementaire {
      Rectangle(int g, int h, int b, int d) {
         Dessin(g, h, b, d);
      }

      void dessineToi () {
         // c'est une primitive
      }
}


class Ligne extends DessinElementaire {
      Ligne(int xp1, int yp1, int xp2, int yp2) {
         Dessin(xp1, yp1, xp2, yp2);
      }

      void dessineToi () {
         // c'est aussi une primitive
      }
}
```

Question A

Donnez le code de la méthode **`dessineToi()`** dans DessinComplex

Question B

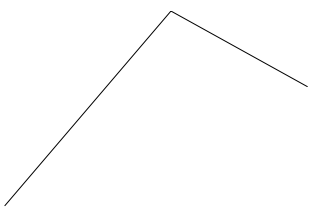
On veut dessiner le dessin complexe **`dc1`** suivant comprenant 2 rectangles et 1 ligne, les rectangles et la ligne étant définis ainsi:

```
Rectangle r1 = new Rectangle(10,10,20,40);
Rectangle r2 = new Rectangle(5,25,50,60);
Ligne l1 = new Ligne(5,35,45,10);
```

Donnez le code de l'instanciation permettant de construire **`dc1`**.

Question C

On désire réaliser la classe DoubleLigne qui comprend 2 lignes reliées entre elles, comme le montre la figure suivante:



Une instance de cette ligne est définie ainsi:

```
DoubleLigne dl1=DoubleLigne(5,10,30,35,20,60);
```

Question: Définissez la classe DoubleLigne avec ses champs et ses méthodes (on donnera le code des méthodes). Note: il existe plusieurs manières de construire une telle classe. Produisez en simplement une.

Question D

On désire calculer le centre d'un dessin. On suppose alors que la classe Dessin est modifiée ainsi (les parties modifiées sont en italique):

```
class Dessin extends Object {
      int x1, y1, x2, y2;
      int cx, cy;        // coordonnées du centre du dessin
      dessineToi() {}
      abstract Entier largeur();    // méthode abstraite
      abstract Entier hauteur();    // méthode abstraite
      calculCentre(){...};
}
```

1) Donnez le code de l'ensemble des méthodes **hauteur()** et **largeur()** dans DessinElementaire et DessinComplexe, de telle manière que l'on puisse calculer la hauteur ou la largeur de n'importe quel dessin. Ex:

```
// calcul la largeur du rectangle r1 c'est-à-dire la différence
// des abscisses des deux points de r1
      r1.largeur();
// calcule la "hauteur" de la ligne, c'est-à-dire la différence
// des ordonnées des deux points de r1
      l1.hauteur();

// calcule la largeur totale du dessin complexe dc1.
      dc1.largeur();
```

2) Donnez le code de la méthode **calculCentre** dans la classe **Dessin**.