UNIVERSITE LIBRE DE BRUXELLES
Faculté des Sciences Appliquées

Juin 2016

INFO-H-2001 – Programmation Orientée-Objet – Examen (3h30)
Hugues Bersini

Remarques préliminaires

- Ne dégraffez pas les feuilles du questionnaire.
- Inscrivez vos nom, prénom et numéro de matricule sur chaque feuille.
- Les notes et calculatrices sont interdites.
- Le projet compte pour 40% de la note finale et l'examen pour 60%.
- La pondération des questions est indiquée. Lisez les questions et choisissez l'odre dans lequel y répondre.

Partie pratique (10 Points)

Question 10 : Syntaxe et Orienté-Objet (4 Point(s))

Corrigez, à même le code, le programme fourni de manière à ce qu'il puisse s'exécuter sans erreur. Indiquez les erreurs qui se produiraient que ce soit à la compilation ou à l'exécution et/ou les éléments manquant. Expliquez brièvement chaque erreur.

```
public class Main {
  private Dog dog;

  public static void main(String[] args) {
    dog = new Dog(0,0,"brown");
    Animal animal = new Animal(5,5,"white");
    Wolf animal1 = new Wolf(1,2,"gray");
    Dog animal2 = new Dog(2,1,"black");
    animal1.eat("meat");
    animal1.howl();
    animal2.eat("sandwich");
    Animal animal3 = animal1;
    Animal animal4 = animal1;
    ((Wolf) animal3).howl();
    ((Wolf)) animal4).howl();
  }
}
```

```
public abstract class Animal {
  private int posX;
  private int posY;
  private String color;
  private boolean hungry;

  public Animal(int x, int y, String color){
    this.posX = x;
    this.posY = y;
    this.color = color;
    this.hungry = true;
  }

  private static void setColor(String color){
    this.color = color;
  }

  public abstract void eat(String food){
    this.hundry = false;
  }

  public abstract void drink();
}
```

```
public class Wolf extends Animal implements Howler{

  public Wolf(int x, int y, String color){
    super.x = x;
    super.y = y;
```

```
    super.color = color;
  }

  public void eat(String food) {
  }

  public void drink() {
  }
}
```

```
public class Dog extends Animal{

  public Dog(int x, int y, String color){
    super(x,y,color);
  }

  public void eat(String food);

  public void drink() {}
}
```

```
public interface Howler {
  public void howl();
}
```

Corrigé

```

public class Main {
  private Dog dog;

  public static void main(String[] args) {
    dog = new Dog(0,0,"brown");         // Dog should be static
    Animal animal = new Animal(5,5,"white");  // Cannot instantiate abstract
    Wolf animal1 = new Wolf(1,2,"gray");
    Dog animal2 = new Dog(2,1,"black");
    animal1.eat("meat");
    animal1.howl();
    animal2.eat("sandwich");
    Animal animal3 = animal1;
    Animal animal4 = animal1;
    ((Wolf) animal3).howl();
    ((Wolf)) animal4).howl();          // animal4 is a dog, not a Wolf
  }

}
```

```

public abstract class Animal {
  private int posX;               // Should be protected for Wolf to work
  private int posY;               // Should be protected for Wolf to work
  private String color;             // Should be protected for Wolf to work
  private boolean hungry;

  public Animal(int x, int y, String color){
    this.posX = x;
    this.posY = y;
    this.color = color;
    this.hungry = true;
  }

  private static void setColor(String color){   // Should not be static considering 'this'
    this.color = color;
  }

  public abstract void eat(String food){      // Should not have a body
    this.hundry = false;
  }

  public abstract void drink();
}
```

```
public class Wolf extends Animal implements Howler{

  public Wolf(int x, int y, String color){
    super.x = x;              // Visilibity problem
    super.y = y;              // Visilibity problem
    super.color = color;          // Visilibity problem
  }
```

```
  // Missing method howl() (interface)

  @Override
  public void eat(String food) {
  }

  @Override
  public void drink() {
  }
}
```

```
public class Dog extends Animal{

  public Dog(int x, int y, String color){
    super(x,y,color);
  }

  @Override
  public void eat(String food);   // Requires a body

  @Override
  public void drink() {}
}
```

```
public interface Howler {
  public void howl();
}
```

Question 11 : Ecriture de code (2 Point(s))

Réalisez dans une syntaxe approximative (mais respectant les principes "Orienté-Objet") un code Java contenant trois classes. Une classe "Homme" avec ses attributs "nom", "prénom" et "age", une classe "Femme" avec les mêmes informations et une classe "Humain". Implémenter aussi les méthodes suivantes :

1. Une méthode "epouser(...)" qui devra permettre à deux personnes de se lier.
2. Une methode "ageDifference()" qui devra calculer la différence d'age entre deux personnes.

Vous pouvez ajouter les attributs et méthodes que vous jugez nécessaire. Votre proposition devra être brièvement justifiée. Enfin, réaliser un diagramme de classe modélisant votre programme.

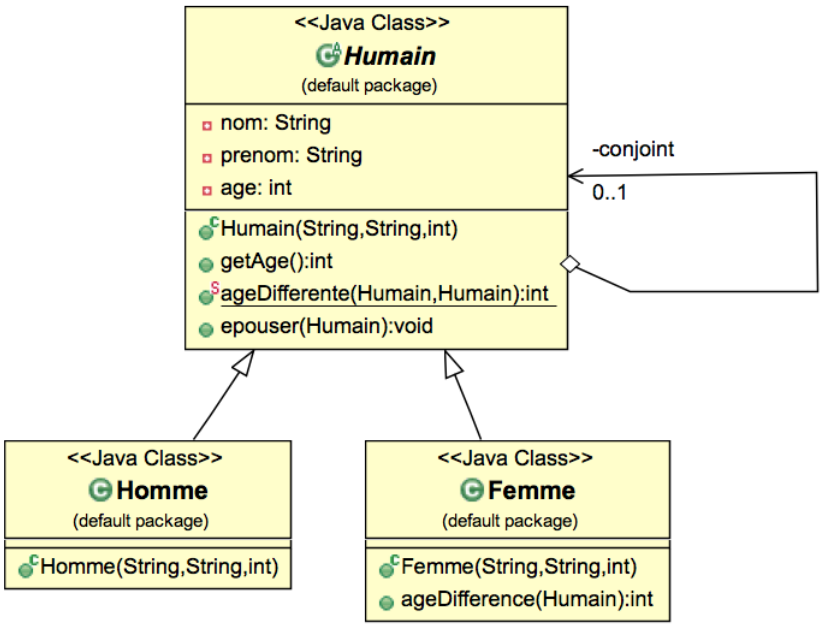
Corrigé

```
public abstract class Humain {
  private String nom;
  private String prenom;
  private int age;
  private Humain conjoint;

  public Humain(String nom, String prenom, int age){
    this.nom = nom;
    this.prenom = prenom;
    this.age = age;
    this.conjoint = null;
  }

  public int getAge(){
    return this.age;
  }

  public static int ageDifferente(Humain h1, Humain h2){
    return Math.abs(h1.getAge() - h2.getAge());
  }

  public void epouser(Humain conjoint){
    this.conjoint = conjoint;
  }
}
```

```
public class Homme extends Humain{

  public Homme(String nom, String prenom, int age) {
    super(nom, prenom, age);
  }
```

```
  public int ageDifference(Humain humain){
    return Math.abs(this.getAge() - humain.getAge());
  }
}
```

```
public class Femme extends Humain{

  public Femme(String nom, String prenom, int age) {
    super(nom, prenom, age);
  }

  public int ageDifference(Humain humain){
    return Math.abs(this.getAge() - humain.getAge());
  }
}
```



Question 12 : Lecture de code et diagrammes UML (4 Point(s))

Le code Java fourni ne présente aucune erreur. Pour ce dernier, veuillez répondre aux questions suivantes :

1. Donner la sortie console de ce programme. (i.e : Ce qui s'imprime à l'écran)
2. Donner le diagramme de classes de ce programme
3. Donner le diagramme de séquence que l'on pourrait générer au départ de l'instruction "drawing.describe()"

```
public class Main {

  public static void main(String[] args) {
    Drawing drawing = new Drawing(0);

    double angle = 45.0;
    for(int i = 0; i < 4; i++){
      Square square = new Square(new Point(5,5), i);
      square.rotate(angle);
      drawing.addFigure(square);

      Circle circle = new Circle(new Point(2,3), i*2);
      circle.translate(i*2, i*2);
      drawing.addFigure(circle);

      angle += 45.0;
    }
    drawing.describe();
  }

}
```

```
import java.util.ArrayList;

public class Drawing {
  private ArrayList<Figure> figures = new ArrayList<Figure>();
  private int backgroundColor;

  public Drawing(int backgroundColor){
    this.backgroundColor = backgroundColor;
  }

  public void addFigure(Figure figure){
    this.figures.add(figure);
  }

  public void setBackgroundColor(int color){
    this.backgroundColor = color;
  }

  public int getBackgroundColor(){
    return this.backgroundColor;
  }

  public ArrayList<Figure> getFigures(){
    return this.figures;
  }

  public void describe(){
    System.out.print("Background color : "+String.valueOf(this.backgroundColor));
    for(Figure figure : figures){
      System.out.println(figure);
    }
  }

}
```

```
public abstract class Figure {

  protected Point center;
  protected double angle;

  public Figure(Point center){
    this.center = center;
    this.angle = 0;
  }

  public Figure(Point center, double angle){
    this.center = center;
    this.angle = angle;
  }
```

```
  public void setAngle(double angle){
    this.angle = angle;
  }

  public void translate(double x, double y) {
    this.center.setPositionX(this.center.getPositionX() + x);
    this.center.setPositionY(this.center.getPositionY() + y);
  }

  public void rotate(double angle){
    this.angle = this.angle + angle;
  }

  public abstract double getPerimeter();
  public abstract double getSurface();
  public abstract String toString();
}
```

```
public class Point {
  private double positionX;
  private double positionY;

  public Point(double x, double y){
    this.positionX = x;
    this.positionY = y;
  }

  public void setPositionX(double x){
    this.positionX = x;
  }

  public void setPositionY(double y){
    this.positionY = y;
  }

  public double getPositionX(){
    return this.positionX;
  }

  public double getPositionY(){
    return this.positionY;
  }

}
```

```
public class Circle extends Figure {

  private double rayon;

  public Circle(Point center, double rayon){
    super(center);
    this.rayon = rayon;
  }

  @Override
  public double getPerimeter() {
    return 2 * Math.PI * this.rayon;
  }

  @Override
  public double getSurface() {
    return Math.PI * this.rayon * this.rayon;
  }

  @Override
  public String toString() {
    return "CIRCLE of radius "+String.valueOf(this.rayon)+ " @ position (x,y) =
         ("+String.valueOf(this.center.getPositionX()+","+String.valueOf(this.center.getPositionY())+")") ;
  }
}
```

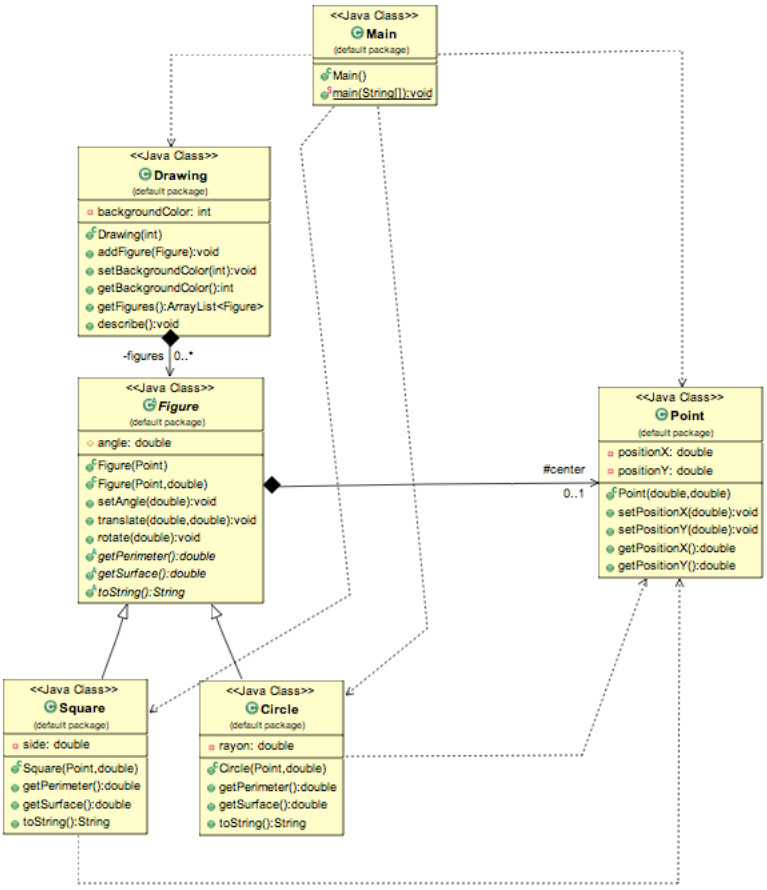
```
public class Square extends Figure{

  private double side;

  public Square(Point center, double side) {
    super(center);
    this.side = side;
  }

  @Override
```

```
  public double getPerimeter() {
    return 4.0 * this.side;
  }

  @Override
  public double getSurface() {
    return this.side * this.side;
  }

  @Override
  public String toString() {
    return "SQUARE of side "+String.valueOf(this.side)+ " @ position (x,y) =
         ("+String.valueOf(this.center.getPositionX()+","+String.valueOf(this.center.getPositionY())+") and
         angle = "+String.valueOf(this.angle));
  }
}
```

Corrigé : Partie 1

Background color : 0
SQUARE of side 0.0 @ position (x,y) = (5.0,5.0) and angle = 45.0
CIRCLE of radius 0.0 @ position (x,y) = (2.0,3.0)
SQUARE of side 1.0 @ position (x,y) = (5.0,5.0) and angle = 90.0
CIRCLE of radius 2.0 @ position (x,y) = (4.0,5.0)
SQUARE of side 2.0 @ position (x,y) = (5.0,5.0) and angle = 135.0
CIRCLE of radius 4.0 @ position (x,y) = (6.0,7.0)
SQUARE of side 3.0 @ position (x,y) = (5.0,5.0) and angle = 180.0
CIRCLE of radius 6.0 @ position (x,y) = (8.0,9.0)

Corrigé : Partie 2



Corrigé : Partie 3