

Nom :

Prénom :

Programmation Orientée Objets - Semestre d'automne 2023

Stéphane Malandain

## Test n°2

30 minutes – 1 formulaire a4 autorisé

### Question 1 (6 pts)

Après avoir mis en commentaire les lignes `System.out.println()` qui provoquent des erreurs de compilation dans le programme suivant, dites ce qu'il affiche. Justifiez chaque mise en commentaire et affichage. Précisez le type déclaré et le type courant de `x`, `y`, `z` et `v`.

```
1  interface I1 { int m1(); }
2
3  interface I2 extends I1 { int m2(); }
4
5  class B implements I1 {
6      int i = 5;
7      public int m1() {
8          return i;
9      }
10 }
11
12 class C extends B implements I2 {
13     int i = -2;
14     public int m1() {
15         return i;
16     }
17     public int m2() {
18         return 2*i;
19     }
20 }
21
22 public class exo1 {
23     Run | Debug
24     public static void main(String[] args) {
25         B x = new C();
26         System.out.println(x.i);
27         System.out.println(x.m1());
28         System.out.println(x.m2());
29         I1 y = x;
30         System.out.println(y.i);
31         System.out.println(y.m1());
32         System.out.println(y.m2());
33         C z = (C)x;
34         System.out.println(z.i);
35         System.out.println(z.m1());
36         System.out.println(z.m2());
37         I2 v = (I2)x;
38         System.out.println(v.i);
39         System.out.println(v.m1());
40         System.out.println(v.m2());
41     }
42 }
```

Nom :

Prénom :

Réponse :

```
public class exo1 {  
    Run | Debug  
    public static void main(String[] args) {  
        B x = new C();  
        System.out.println(x.i);  
        System.out.println(x.m1());  
        // System.out.println(x.m2());  
        I1 y = x;  
        // System.out.println(y.i);  
        System.out.println(y.m1());  
        // System.out.println(y.m2());  
        C z = (C)x;  
        System.out.println(z.i);  
        System.out.println(z.m1());  
        System.out.println(z.m2());  
        I2 v = (I2)x;  
        // System.out.println(v.i);  
        System.out.println(v.m1());  
        System.out.println(v.m2());  
    }  
}
```

	Type déclaré	type courant
x	B	C
y	I1	C
z	C	C
v	I2	C

Comme les types déclarés de *x*, *y* et *v* sont respectivement *B*, *I1* et *I2*, le compilateur ne trouve pas de méthode *m2()* pour *x* et *y*, ni de champ *i* pour *y* et *v*. Le type courant étant toujours *C*, les méthodes *m1()* et *m2()* retournent respectivement -2 et -4. Par contre, c'est le type déclaré qui compte pour la variable d'instance *i*.

Affichage:

```
5  
-2  
-2  
-2  
-2  
-4  
-2  
-4
```

Nom :

Prénom :

### Question 3 – Classe abstraite (10 pts)

3.1. Définir une classe abstraite, nommé `FormeGeometrique.java` qui a les caractéristiques suivantes :

- ☐ 2 coordonnées (double) `x, y` qui représentent le centre de la forme
- ☐ Une méthode `deplacer()` qui effectue une translation de la forme de `dx` et `dy`.
- ☐ Une méthode `AfficherPosition` qui affiche la position `x, y` de la forme.
- ☐ Une méthode abstraite `Surface` qui renvoie la surface de la figure géométrique.

Code ici :

```
1  public abstract class FormeGeometrique {
2      protected double x = 0.0;
3      protected double y = 0.0;
4
5      public FormeGeometrique( double x, double y) {
6          this.x = x;
7          this.y = y;
8      }
9
10     public void deplacer (double dx, double dy) {
11         x += dx;
12         y += dy;
13     }
14     public void afficherPosition() {
15         System.out.println("position : (" + x + " , " + y + " )");
16     }
17     public abstract double surface();
18
19 }
```

3.2. Cette classe est-elle instanciable ? *Non*

3.3. Définir une classe concrète qui hérite de la classe `Figure`, nommée `Circle.java`. Ajouter le ou les attributs nécessaires pour créer un cercle.

Code ici :

Nom :

Prénom :

```
users / mairan / Documents / nepia / java / 2022 / tests / test2
1  public class Circle extends FormeGeometrique {
2      protected double rayon ;
3
4      public Circle(double x, double y, double r) {
5          super(x,y);
6          this.rayon=r;
7      }
8
9      public double surface() {
10         return 3.14*rayon*rayon;
11     }
12 }
```

#### Question 4 - Classe, interface et polymorphisme (10 pts)

Définir une interface **Additionnable** constituée d'une seule méthode **somme** qui prend en paramètre un **Object** et retourne un **Object**.

Un nombre rationnel est caractérisé par son numérateur et son dénominateur. Soit les deux nombres rationnels  $r1=n1/d1$  et  $r2=n2/d2$ . La somme de deux nombres rationnels est définie comme :  $r1+r2=((n1*d2)+(n2*d1))/(d1*d2)$ , avec  $d1$  et  $d2$  non nuls.

Ecrire la classe **Rationnel** qui décrit des nombres rationnels :

- ☐ avec des variables d'instances ;
- ☐ avec au moins un constructeur permettant l'appel

```
Rationnel r = new Rationnel(n,d);
// n et d des integer, d non nul.
```

- ☐ en redéfinissant la méthode nécessaire pour obtenir un **String** ;
- ☐ en implémentant l'interface **Additionnable**;
- ☐ en surchargeant la méthode **somme** pour pouvoir faire l'appel

```
Rationnel r = Rationnel.somme(r1,r2) ;
// r1, r2 des rationnels.
```

Code ici :

Nom :

Prénom :

```
1  interface Additionnable {
2      public Object somme(Object o);
3  }
4
5  public class Rationnel implements Additionnable {
6      private int n=0;
7      private int d=1;
8
9      public Rationnel(int x, int y) {
10         n=x;
11         if (y!=0) d=y;
12     }
13
14     public String toString() { return n+"/"+d;}
15
16     public Object somme(Object o) {
17         if (o instanceof Rationnel) {
18             Rationnel c = (Rationnel)o;
19             return new Rationnel(this.n*c.d+this.d*c.n, this.d*c.d);
20         } else return null;
21     }
22
23     static public Rationnel somme(Rationnel r1, Rationnel r2) {
24         if (r1 != null) return r1.somme(r2);
25         else return null;
26     }
27 }
28
```