

Nom :

Prénom :

Programmation Orientée Objet - Semestre d'automne 2024
Stéphane Malandain / Yassin Rekik

Test n°2

40 minutes – 1 formulaire a4 autorisé

Question 1 (2 pts)

Les exemples suivants sont-ils corrects (compilation – exécution) ? Justifiez

Exemple 1.1

```
1 class A{
2     private int a;
3     A(int a) {this.a = a;}
4 }
5
6 class B extends A{
7     private int b;
8     B() { this.a = 0; this.b = 0;}
9 }
```

Réponse 1.1 :

Non : la classe B n'a pas accès au champ a. il faut utiliser le constructeur de la super classe super() ; ensuite this.b = 0 ;

Exemple 1.2

```
1 class A{
2     public int f(int a) { return a++; }
3 }
4
5 class B extends A{
6     public int f(int a, int b) { return a+b; }
7 }
8
9 class Test {
10     Run | Debug
11     public static void main(String[] args) {
12         A objet1 = new B();
13         int x = objet1.f(a:3);
14         int y = objet1.f(3,3);
15     }
```

Réponse 1.2 :

Non, le compilateur ne va pas compiler car objet1 a été déclaré de la classe A pour lequel f est défini avec un seul argument.

Nom :

Prénom :

Question 2 : polymorphisme (3 pts)

2.1 Qu'affiche le programme suivant ?

```
1  class A{
2      public String f(D obj) {return "A et D";}
3      public String f(A obj) {return "A et A";}
4  }
5
6  class B extends A{
7      public String f(B obj) {return "B et B";}
8      public String f(A obj) {return "B et A";}
9  }
10
11 class C extends B{
12 }
13 class D extends B{
14 }
15
16 class Test3{
17     public static void main (String[] args) {
18         A a1=new A();
19         A a2=new B();
20         B b=new B();
21         C c=new C();
22         D d=new D();
23         System.out.println(a1.f(b));
24         System.out.println(a1.f(c));
25         System.out.println(a1.f(d));
26         System.out.println(a2.f(b));
27         System.out.println(a2.f(c));
28         System.out.println(a2.f(d));
29         System.out.println(b.f(b));
30         System.out.println(b.f(c));
31         System.out.println(b.f(d));
32     }
33 }
```

Réponse :

```
A et A
A et A
A et D
B et A
B et A
A et D
B et B
B et B
A et D
```

Les premiers trois appels se comportent d'une façon polymorphe comme on s'y attend.

Les trois appels suivants semblent un peu bizarres cela dépend du fait que la signature de la méthode est choisie à la compilation, tandis que le code à exécuter est choisi à l'exécution.

A la compilation, l'objet a2 est supposé être de type A . Dans la classe A, la méthode f n'est pas définie pour un objet de type B . Ainsi, l'appel a2.f(b) utilise le polymorphisme et considère b comme un objet de type A. Donc, la signature pour cet appel exige un argument de type A. A l'exécution, a2 est un objet de type B. Le code utilisé est celui de la méthode de B qui a la même signature, même si B pourrait avoir une méthode plus précise. L'appel a2.f(d) choisit la signature de f qui exige objets de type D. A l'exécution, a2 est de type B. La classe B n'a pas une méthode f avec la bonne signature. Donc le code de la superclasse est utilisé.

Nom :

Prénom :

Question 3 (4 pts)

```
1  class Parent {
2      public int foo(){
3          return 0;
4      }
5  }
6
7  class Fils extends Parent {
8      public int goo(){
9          return 1;
10     }
11 }
12
13 class Fille extends Parent {
14     public int goo(){
15         return 2;
16     }
17 }
18
19 public class Main {
20     Run | Debug
21     public static void main(String[] args) {
22         Parent[] Famille = new Parent[4];
23         Famille[0] = new Fils();
24         Famille[1] = new Fille();
25         Famille[2] = new Fils();
26         Famille[3] = new Fille();
27         for (Parent p : Famille){
28             System.out.println(p.goo());
29             System.out.println(p.foo());
30         }
31     }
32 }
```

3.1 Est-ce que le code précédent compile ? si non, pourquoi ?

Réponse :

Non le code ne compile car dans parent pas de méthode `parent.goo()`

3.2. En agissant **uniquement** sur la classe `Parent`, et sachant que la classe `Parent` ne peut pas décider de l'implémentation de la méthode `goo()`, pouvez-vous proposer une solution pour résoudre le problème et faire que le code compile et fonctionne.

```
abstract class Parent {
    public int foo(){
        return 0;
    }
    public abstract int goo();
}
```

Nom :

Prénom :

Question 4 - Classe, héritage et polymorphisme (12 pts)

4.1. Définir une classe abstraite, nommé `Personne` qui a les caractéristiques suivantes :

- Un nom de type `String`
- Un âge représenté par un entier
- Un genre représenté par un caractère
- Une méthode `AfficherPersonne` qui affiche le nom de la personne, son âge et son genre.
- Une méthode abstraite `getActivite` qui renvoie le domaine d'activité d'une personne : un `String` représentant le domaine d'activité comme Informatique, Math, Architecture,

Code ici :

```
1  public abstract Personne {
2      protected String nom;
3      protected int age;
4      protected char genre;
5
6      public AfficherPersonne() {
7          System.out.println(nom + " " + age + " " + genre);
8      }
9      public abstract String getActivite();
10 }
```

4.2. Cette classe est-elle instanciable ? Pourquoi ?

Non car c'est une classe abstraite

Nom :

Prénom :

4.3. Ecrire deux interfaces :

4.3.1 Travailleur qui propose 2 méthodes :

- `getSalaire` qui retourne le salaire d'une personne.
- `changeSalaire` avec un entier en paramètre qui modifie la valeur du salaire;

4.3.2 Sportif qui propose 1 méthode :

- `getSport` qui retourne le sport pratiqué par une personne.

Code ici :

```
12 public interface Travailleur {
13     public int getSalaire();
14     public void changeSalaire(int montant);
15 }
16
17 public interface Sportif {
18     public String getSport();
19 }
20
```

3.4. Définir 3 classes concrètes

Etudiant qui hérite de la classe `Personne`,

EtudiantSportif qui hérite de la classe `Personne` et qui implémente l'interface `Sportif`.

EnseignantSportif qui hérite de la classe `Personne` et qui implémente les interfaces `Sportif` et `Travailleur`.

.

Code ici :

```
21 public class Etudiant extends Personne {
22     protected String activite;
23
24     public String getActivite() { return Activite; }
25 }
26
27 public class EtudiantSportif extends Etudiant implements Sportif {
28     protected String sport;
29
30     public String getSport() { return sport; }
31 }
32
33 public class EnseignantSportif extends Personne implements Sportif, Travailleur {
34     protected String activite;
35     protected String sport;
36     protected int salaire;
37
38     public String getActivite() { return Activite; }
39     public String getSport() { return sport; }
40     public void changeSalaire(int montant) { Salaire += montant; }
41 }
42
```

Nom :

Prénom :