

Tous documents interdits.

NOM :

PRÉNOM :

▷ **Question 1.** (4 pt)

Indiquer si chaque affirmation est correcte ou non :

Vrai	Faux	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Un variable d'instance marquée comme private est accessible uniquement depuis l'instance de l'objet concerné.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	L'instruction <code>(new String("abc")).toUpperCase() == (new String("ABC"))</code> est évaluée à vraie.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	L'instruction <code>(new Integer(1)).equals(new Integer(1))</code> est évaluée à vraie.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Il n'est pas toujours possible de définir une sous-classe d'une classe existante.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Une classe peut hériter de plusieurs classes.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Une classe peut avoir deux méthodes dont les profils sont identiques hormis le type de retour.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Une sous-classe hérite de toutes les variables d'instance et de toutes les méthodes de sa super-classe.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Si dans un programme Java vous lisez <code>X implements Y</code> alors <code>Y</code> est obligatoirement une interface.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Si dans un programme Java vous lisez <code>X extends Y</code> alors <code>Y</code> est obligatoirement une classe.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Seule une méthode marquée static peut modifier une variable de classe.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	En tant que développeur, vous marquez une méthode abstract pour obliger sa définition dans les sous-classes.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Dans une classe, on ne peut définir qu'au plus 64 méthodes à deux paramètres portant le même nom.

▷ **Question 2.** (1 pt)

Considérer les classes suivantes et indiquer la réponse correcte :

TestCounter.java

```

1 class Counter {
2     private static int counter = 0;
3
4     public void increment() {
5         Counter.counter++;
6     }
7     public int getValue() {
8         return counter;
9     }
10 }
11
12 class TestCounter {
13     public static void main(String args[]) {
14         Counter c1 = new Counter();
15         c1.increment();
16         Counter c2 = new Counter();
17         c2.increment();
18
19         System.out.println(c2.getValue());
20     }
21 }
```

- ☐ Ce code ne compile pas (erreur ligne 2).
- ☐ Ce code ne compile pas (erreur ligne 5 et 8).
- ☐ Ce code ne compile pas (erreur lignes 15 et 17).
- ☐ Ce code compile et le programme affiche : 0
- ☐ Ce code compile et le programme affiche : 1
- ☒ Ce code compile et le programme affiche : 2

▷ **Question 3.** (1 pt)

Considérer l'extrait de code Java suivant. Quel sera l'affichage produit par l'exécution de ce code ?

```

1 class Mere {
2     void a() {
3         System.out.println("mere - a");
4     }
5
6     public void b() {
7         System.out.println("mere - b");
8     }
9 }
10
11 class Fille extends Mere {
12     public void a() {
13         System.out.println("fille - a");
14     }
15
16     public void b() {
17         System.out.println("fille - b");
18     }
19
20     public static void main(String args[]) {
21         Mere o = new Fille();
22         o.a();
23         o.b();
24     }
25 }

```

- ☒ fille - a , fille - b.
- ☐ mere - a , fille - b.
- ☐ Une erreur à la compilation est détectée ligne 21.
- ☐ Une erreur à la compilation est détectée ligne 22.
- ☐ Une erreur à la compilation est détectée ligne 23.

▷ **Question 4.** (1 pt)

Considérer l'extrait de code Java suivant :

```

1 String example = "Hi girls!";
2 System.out.print(example.toUpperCase().charAt(3));

```

Vrai

☒

Faux

☐

L'exécution de ce programme affiche : G

▷ **Question 5.** (2 pt)

Déterminer l'affichage en sortie standard lors de l'exécution de la méthode `filter(int x)`, en fonction de la valeur entière `x` passée en paramètre.

```

1 void filter(int x) {
2     System.out.print("a");
3     try {
4         System.out.print("b");
5         try {
6             System.out.print("c");
7             if (x < 0)
8                 throw new FirstException();
9             System.out.print("d");
10            if (x == 0)
11                throw new SecondException();
12            System.out.print("e");
13            if (x > 0)
14                System.out.print("f");
15            System.out.print("g");
16        } catch (FirstException e) {
17            System.out.print("h");
18        } finally {
19            System.out.print("i");
20        }
21        System.out.print("j");
22    } catch (SecondException e) {
23        System.out.print("k");
24    } finally {
25        System.out.print("l");
26    }
27    System.out.print("m");
28 }
29
30 class FirstException extends Exception {};
31 class SecondException extends Exception {};

```

`x < 0` : abchijlm
`x = 0` : abcdiklm
`x > 0` : abcdefgijlm

▷ **Question 6.** (2 pt)

Considérer la classe suivante et indiquer **la/les réponses correcte(s)** si l'instruction proposée est insérée en ligne 30. Si il n'y a pas d'erreur, préciser l'affichage.

```
1 class Fruit {
2     protected int id;
3
4     public Fruit(int myid) {
5         this.id = myid;
6     }
7     public boolean equals(Fruit a) {
8         return (a.id == this.id);
9     }
10 }
11
12 class Apple extends Fruit {
13     private String name;
14     public Apple(int myid, String name) {
15         super(myid);
16         this.name = name;
17     }
18     public boolean equals(Apple d) {
19         return (d.id == this.id
20             && this.name.equals(d.name));
21     }
22 }
23
24 class Main {
25     public static void main(String args[]) {
26         boolean b = false;
27         Apple golden = new Apple(29, "Golden");
28         Apple red = new Apple(29, "Red");
29         Fruit green = new Apple(29, "Green");
30         // instruction a inserer
31         System.out.println(b);
32     }
33 }
```

	Erreur	Ok	Affichage
b = golden.equals(green);	<input type="checkbox"/>	<input checked="" type="checkbox"/>	true
b = golden.equals(red);	<input type="checkbox"/>	<input checked="" type="checkbox"/>	false
b = green.equals(golden);	<input type="checkbox"/>	<input checked="" type="checkbox"/>	true
b = red.equals(golden);	<input type="checkbox"/>	<input checked="" type="checkbox"/>	false

▷ **Question 7.** Schéma mémoire (4 pt)

Considérer les classes suivantes et réaliser des schémas de la mémoire (états de la pile et du tas) lors de l'exécution de la méthode `main()` de la classe principale `Main` aux points identifiés dans le code source (indiqués par le marqueur `// POINT_? schema mémoire à cet instant`).

Vous préciserez l'affichage obtenu sur la sortie standard à chaque étape.

```

1 interface Expr {
2     int evaluer() ;
3     String decompiler();
4 }
5
6 class Value implements Expr {
7     private int val;
8
9     public Value(int v) {
10         this.val = v;
11     }
12
13     public int evaluer() {
14         return this.val;
15     }
16
17     public String decompiler() {
18         return ""+this.val;
19     }
20 }
21
22 class Add implements Expr {
23     private Expr g;
24     private Expr d;
25
26     public Add(Expr g, Expr d) {
27         this.g = g;
28         this.d = d;
29     }
30
31     public void setG(Expr g) {
32         this.g = g;
33     }
34
35     public void setD(Expr d) {
36         this.d = d;
37     }
38
39     public int evaluer() {
40         return this.g.evaluer() + this.d.evaluer();
41     }
42
43     public String decompiler() {
44         return "("+this.g.decompiler()+
45             "+"+this.d.decompiler()+")";
46     }
47 }

```

```

1 class Main {
2     public static void main(String args[]) {
3         Expr v1 = new Value(42);
4         Expr v2 = new Value(17);
5
6         Expr a = new Add(v1, v2);
7         // POINT_1 schema mémoire à cet instant
8         System.out.println(v1.decompiler() + " = " + v1.evaluer());
9         System.out.println(v2.decompiler() + " = " + v2.evaluer());
10        System.out.println(a.decompiler() + " = " + a.evaluer());
11
12        v2 = v1;
13        v1 = new Value(13);
14        // POINT_2 schema mémoire à cet instant
15        System.out.println(v1.decompiler() + " = " + v1.evaluer());
16        System.out.println(v2.decompiler() + " = " + v2.evaluer());
17        System.out.println(a.decompiler() + " = " + a.evaluer());
18
19        a.setG(v1);
20        // POINT_3 schema mémoire à cet instant
21        System.out.println(v1.decompiler() + " = " + v1.evaluer());
22        System.out.println(v2.decompiler() + " = " + v2.evaluer());
23        System.out.println(a.decompiler() + " = " + a.evaluer());
24
25        a.setD(a);
26        // POINT_4 schema mémoire à cet instant
27        System.out.println(v1.decompiler() + " = " + v1.evaluer());
28        System.out.println(v2.decompiler() + " = " + v2.evaluer());
29        System.out.println(a.decompiler() + " = " + a.evaluer());
30    }
31 }

```

Réponse

```

1 // POINT_1
2     42 = 42
3     17 = 17
4     (42+17) = 59
5 // POINT_2
6     13 = 13
7     42 = 42
8     (42+17) = 59
9 // POINT_3
10    13 = 13
11    42 = 42
12    (13+17) = 30
13 // POINT_4
14    13 = 13
15    42 = 42
16 Exception in thread "main" java.lang.StackOverflowError
17   at java.util.Arrays.copyOfRange(Arrays.java:3210)
18   at java.lang.String.<init>(String.java:215)
19   at java.lang.StringBuilder.toString(StringBuilder.java:430)
20   at Value.decompiler(Expr.java:18)
21   at Add.decompiler(Expr.java:44)
22   at Add.decompiler(Expr.java:44)

```

Fin réponse