

NOM :

PRÉNOM :

▷ **Question 1.** (4 pt)

Indiquer si l'affirmation est correcte ou non :

Vrai	Faux	
<input type="checkbox"/>	<input type="checkbox"/>	Une variable d'instance est accessible depuis l'extérieur d'une instance lorsqu'elle est déclarée public .
<input type="checkbox"/>	<input type="checkbox"/>	Seule une méthode marquée static peut modifier une variable de classe.
<input type="checkbox"/>	<input type="checkbox"/>	Une classe peut avoir deux méthodes dont les profils sont identiques hormis le type de retour.
<input type="checkbox"/>	<input type="checkbox"/>	Comme la méthode <code>parseInt</code> de la classe <code>Integer</code> peut être invoquée par <code>Integer.parseInt("12")</code> , il s'agit obligatoirement une méthode de classe.
<input 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 type="checkbox"/>	<input type="checkbox"/>	Le <i>surcharge</i> permet la présence d'une méthode ayant le même nom avec des profils différents dans la même classe.
<input type="checkbox"/>	<input type="checkbox"/>	Une variable d'instance marquée private est accessible dans les sous-classes.
<input 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 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 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 type="checkbox"/>	Lorsqu'une exception est levée, plusieurs blocs catch peuvent être exécutés.
<input type="checkbox"/>	<input type="checkbox"/>	Dès qu'une exception est levée, l'exécution du bloc try englobant est arrêté.

▷ **Question 2.** (2 pt)

En considérant les paires de déclarations de méthodes suivantes, indiquez si les affirmations suivantes sont correctes ou non.

<pre>1 void fly(int distance) { 2 } 3 int fly(int time, int speed) { 4 return time*speed; 5 } 6 7 void fall(int time) { 8 } 9 int fall(int distance) { 10 return distance; 11 } 12 13 void glide(int time) { 14 } 15 void Glide(int time) { 16 }</pre>	<table><tr><th>Vrai</th><th>Faux</th><th></th></tr><tr><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td>La première paire de méthodes compilera et surchargera la méthode <code>fly</code>.</td></tr><tr><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td>La deuxième paire de méthodes compilera et surchargera la méthode <code>fall</code>.</td></tr><tr><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td>La deuxième paire de méthodes ne compilera pas.</td></tr><tr><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td>La troisième paire de méthodes compilera et surchargera la méthode.</td></tr><tr><td><input type="checkbox"/></td><td><input type="checkbox"/></td><td>La troisième paire de méthodes ne compilera pas.</td></tr></table>	Vrai	Faux		<input type="checkbox"/>	<input type="checkbox"/>	La première paire de méthodes compilera et surchargera la méthode <code>fly</code> .	<input type="checkbox"/>	<input type="checkbox"/>	La deuxième paire de méthodes compilera et surchargera la méthode <code>fall</code> .	<input type="checkbox"/>	<input type="checkbox"/>	La deuxième paire de méthodes ne compilera pas.	<input type="checkbox"/>	<input type="checkbox"/>	La troisième paire de méthodes compilera et surchargera la méthode.	<input type="checkbox"/>	<input type="checkbox"/>	La troisième paire de méthodes ne compilera pas.
Vrai	Faux																		
<input type="checkbox"/>	<input type="checkbox"/>	La première paire de méthodes compilera et surchargera la méthode <code>fly</code> .																	
<input type="checkbox"/>	<input type="checkbox"/>	La deuxième paire de méthodes compilera et surchargera la méthode <code>fall</code> .																	
<input type="checkbox"/>	<input type="checkbox"/>	La deuxième paire de méthodes ne compilera pas.																	
<input type="checkbox"/>	<input type="checkbox"/>	La troisième paire de méthodes compilera et surchargera la méthode.																	
<input type="checkbox"/>	<input type="checkbox"/>	La troisième paire de méthodes ne compilera pas.																	

▷ **Question 3.** (1 pt)

Considérer les classes suivantes et indiquer la réponse correcte :

<pre> 1 class Counter { 2 private static int counter = 0; 3 4 public void increment() { 5 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 } </pre>	<p>TestCounter.java</p> <ul style="list-style-type: none"> <input type="checkbox"/> Ce code ne compile pas (erreur ligne 2). <input type="checkbox"/> Ce code ne compile pas (erreur ligne 5 et 8). <input type="checkbox"/> Ce code ne compile pas (erreur lignes 15 et 17). <input type="checkbox"/> Ce code compile et le programme affiche : 0 <input type="checkbox"/> Ce code compile et le programme affiche : 1 <input type="checkbox"/> Ce code compile et le programme affiche : 2
---	--

▷ **Question 4.** (2 pt)

Considérer la classe suivante et indiquer **la/les réponses correcte(s)** si l'instruction proposée est insérée en ligne 24. Si il n'y a pas d'erreur, préciser l'affichage.

```

1  class Pokemon {
2      public void chuchu() {
3          System.out.println("Pokemon fait chuchu");
4      }
5      public static void pikapika() {
6          System.out.println("Pokemon fait pikapika");
7      }
8  }
9
10 class Pikachu extends Pokemon {
11     public void chuchu() {
12         System.out.println("Pikachu fait chuchu");
13     }
14     public static void pikapika() {
15         System.out.println("Pikachu fait pikapika");
16     }
17 }
18
19 class PolymorphismTest {
20     public static void main(String[] args) {
21         Pokemon m = new Pokemon();
22         Pokemon b = new Pikachu();
23         // instruction
24     }
25 }
26

```

m.chuchu();

Erreur ?

☐

Si Ok, Affichage

b.chuchu();

☐

((Pikachu) b).chuchu();

☐

m.pikapika();

☐

b.pikapika();

☐

((Pikachu)

☐

b).pikapika();

▷ **Question 5.** (2 pt)

Considérer l'extrait de code Java suivant :

```

1  int speedLimit;
2  ...
3  try {
4      System.out.println("Entering the try block.");
5      if (speedLimit > 130) {
6          throw new Exception("Speed limit violation.");
7      }
8      System.out.println("Exiting the try block.");
9  } catch (Exception e) {
10     System.out.println("Exception: " + e.getMessage());
11 } finally {
12     System.out.println("Inside finally block.");
13 }
14
15 System.out.println("After the catch block.");

```

Quelle est l'affichage si la valeur de la variable **speedLimit** est 50 ?

Quelle est l'affichage si la valeur de la variable **speedLimit** est 150 ?

▷ **Question 6.** (2 pt)

Expliquez le concept de *liaison dynamique* et donner une exemple illustrant ce concept.

▷ **Question 7.** (2 pt)

Expliquez la différence de sémantique et de comportement de l'opérateur `==` et de la méthode `boolean equals(Object o)` présente dans la classe `Object`.

▷ **Question 8.** (2 pt)

Expliquez le principe d'*encapsulation* en programmation orientée-objet et donner un exemple illustrant ce concept.

▷ **Question 9.** Schéma mémoire (3 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 `Test` aux points identifiés dans le code source (indiqués par le marqueur `// point_? schema memoire`).

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

```

1  public class Point {
2      public int a1;
3      public int a2;
4
5      public Point(int u, int v) {
6          a1=u;
7          a2=v;
8      }
9
10     public void translate(int u, int v) {
11         a1 = a1 + u;
12         a2 = a2 + v;
13     }
14
15     public Point copy(int u, int v) {
16         return new Point(a1+u, a2+v);
17     }
18
19     public boolean egale(Point other) {
20         return ( (this.a1 == other.a1)
21                 && (this.a2 == other.a2));
22     }
23
24     public String toString() {
25         return "("+ a1 + ","+ a2+")";
26     }
27 }

```

```

1  public class Triangle {
2      public Point b1;
3      public Point b2;
4      public Point b3;
5
6      public Triangle(Point x, Point y, Point z) {
7          b1=x;
8          b2=y;
9          b3=z;
10     }
11
12     public void translate(int u, int v) {
13         b1.translate(u,v);
14         b2.translate(u,v);
15         b3.translate(u,v);
16     }
17
18     public Triangle copy(int u, int v) {
19         return new Triangle(b1.copy(u,v),
20                             b2.copy(u,v),
21                             b3.copy(u,v));
22     }
23
24     public Triangle duplicate(int u, int v) {
25         b1.translate(u,v);
26         b2.translate(u,v);
27         b3.translate(u,v);
28         return new Triangle(b1,b2,b3);
29     }
30
31     public boolean egale(Triangle other) {
32         return ( (this.b1.egale(other.b1) &&
33                 (this.b2.egale(other.b2) &&
34                 (this.b3.egale(other.b3) ));
35     }
36
37     public String toString() {
38         return "["+ b1 + ";" + b2+";"+b3+"]" ;
39     }
40 }

```

```

1  public class Test {
2
3
4      public static void main(String[] args) {
5          int n1;
6          int n2;
7          Point p1, p2, p3;
8          Triangle f1, f2, f3;
9
10         n1 = 1;
11         n2 = 7;
12         p1 = new Point(n1,n2);
13         System.out.println("p1="+p1);
14         p2 = p1.copy(28,11);
15         p1.translate(28,11);
16         System.out.println("p1="+p1);
17         System.out.println("p2 (ou p1)="+p2);
18         System.out.println(p1==p2);
19         System.out.println(p1.egale(p2));
20         n1 = n1 + n2;
21         n2 = n1 + n2;
22         p3 = new Point (n1, n2);
23         System.out.println("p3 (ou p2)="+p3);
24         p3.translate(21,3);
25         System.out.println("p3 (ou p2)="+p3);
26         System.out.println(p1 == p3);
27         System.out.println(p1.egale(p3));
28         p1=p2;
29         System.out.println(p1==p2);
30         System.out.println(p1.egale(p2));
31         System.out.println("");
32
33         // point_1 schema memoire
34         System.out.println("---- POINT 1 ----");
35
36         p1 = new Point(1,4);
37         p2 = new Point(5,7);
38         p3 = new Point(8,9);
39         f1 = new Triangle(p1,p2,p3);
40         System.out.println("f1="+f1);
41         f1.translate(10,20);
42         System.out.println("f1="+f1);
43         f2 = f1.copy(5,10);
44         System.out.println("f1="+f1);
45         System.out.println("f2="+f2);
46         System.out.println(f1==f2);
47         f1.translate(5,10);
48         System.out.println("f1="+f1);
49         System.out.println(f1==f2);
50         System.out.println(f1.egale(f2));
51         System.out.println("");
52
53         // point_2 schema memoire
54         System.out.println("---- POINT 2 ----");
55
56         p1 = new Point(2,4);
57         p2 = new Point(3,6);
58         p3 = new Point(4,8);
59         f1 = new Triangle(p1,p2,p1);
60         f2 = new Triangle(new Point(2,4),
61                             new Point(3,6),
62                             new Point(2,4));
63         System.out.println(f1==f2);
64         System.out.println(f1.egale(f2));
65         System.out.println("f1="+f1);
66         f1.translate(10,20);
67         System.out.println("f1="+f1);
68         System.out.println("f2="+f2);
69         f3 = f2.duplicate(100,200);
70         System.out.println("f2="+f2);
71         System.out.println("f3="+f3);
72         System.out.println(f2==f3);
73         System.out.println(f2.egale(f3));
74
75         // point_3 schema memoire
76         System.out.println("---- POINT 3 ----");
77     }
78 }

```