

NOM :

PRÉNOM :

### Question 1. (5 pt)

Indiquer si l'affirmation est correcte ou non :

Vrai	Faux	
<input type="checkbox"/>	<input type="checkbox"/>	Deux classes peuvent hériter de la même classe mère.
<input type="checkbox"/>	<input type="checkbox"/>	En Java, la méthode qui sera appelée est toujours déterminée à la compilation.
<input type="checkbox"/>	<input type="checkbox"/>	En Java, seul le type dynamique d'une référence est utilisée pour déterminer la méthode à appeler.
<input type="checkbox"/>	<input type="checkbox"/>	Une classe abstraite ( <b>abstract</b> ) peut être déclarée ( <b>final</b> ).
<input type="checkbox"/>	<input type="checkbox"/>	L'opérateur de transtypage ( <i>cast</i> ) permet uniquement de changer le type statique d'une référence.
<input type="checkbox"/>	<input type="checkbox"/>	Une classe peut implémenter deux interfaces définissant la même méthode.
<input type="checkbox"/>	<input type="checkbox"/>	En Java, l'encapsulation est une encapsulation de classe.
<input type="checkbox"/>	<input type="checkbox"/>	Le receveur d'un appel de méthode peut être de type primitif.
<input type="checkbox"/>	<input type="checkbox"/>	La classe <b>Object</b> est une sous-classe de toutes les classes que l'on peut définir.
<input type="checkbox"/>	<input type="checkbox"/>	Une classe dont tous les champs sont déclarés <b>private</b> ne peut pas être déclarée <b>public</b> .

### Question 2. (2 pt)

Parmi les déclarations de classe et d'interface suivantes, indiquer lesquelles sont valides (certaines déclarations seront ré-utilisées par les déclarations qui les suivent) :

- |  |   |
|--|---|
| (a) <input type="checkbox"/> <code>class Foo { }</code>                    | (h) <input type="checkbox"/> <code>interface Zoo extends Foo { }</code>           |
| (b) <input type="checkbox"/> <code>class Bar implements Foo { }</code>     | (i) <input type="checkbox"/> <code>interface Boo extends Fi { }</code>            |
| (c) <input type="checkbox"/> <code>interface Baz { }</code>                | (j) <input type="checkbox"/> <code>class Zoom implements Fi, Baz { }</code>       |
| (d) <input type="checkbox"/> <code>interface Fi { }</code>                 | (k) <input type="checkbox"/> <code>class Toon extends Foo, Zoom { }</code>        |
| (e) <input type="checkbox"/> <code>interface Fee implements Baz { }</code> | (l) <input type="checkbox"/> <code>interface Vroom extends Fi, Baz { }</code>     |
| (f) <input type="checkbox"/> <code>interface Zee implements Foo { }</code> | (m) <input type="checkbox"/> <code>class Yow extends Foo implements Fi { }</code> |

### Question 3. (1 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 Animal {
2     protected long uniqueId;
3
4     public Animal(long myid) {
5         this.uniqueId = myid;
6     }
7     public boolean equals(Animal a) {
8         return (a.uniqueId == this.uniqueId);
9     }
10 }
11
12 class Duck extends Animal {
13     private String name;
14     public Duck(long myid, String name) {
15         super(myid);
16         this.name = name;
17     }
18     public boolean equals(Duck d) {
19         return (d.uniqueId == this.uniqueId
20             && this.name.equals(d.name));
21     }
22 }
23
24 class Main {
25     public static void main(String args[]) {
26         boolean flag = false;
27         Duck donald = new Duck(1L, "Donald");
28         Duck cloneA = new Duck(1L, "CloneA");
29         Animal cloneB = new Duck(1L, "CloneB");
30         // instruction à insérer
31         System.out.println(flag);
32     }
33 }

```

	Erreur	Ok	Affichage
<code>flag = cloneB.equals(donald);</code>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<code>flag = donald.equals(cloneB);</code>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<code>flag = cloneA.equals(donald);</code>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
<code>flag = donald.equals(cloneA);</code>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

▷ **Question 4.** (1 pt)

Considérer la classe suivante et indiquer quelle est/sont la ou les erreur(s) qui empêche(nt) la compilation :

```

1 class Donut {
2 }
3
4 class DonutFactory {
5     int donutCount = 0;
6     public DonutFactory() {
7     }
8
9     public static Donut cookDonut() {
10         donutCount = donutCount + 1;
11         return new Donut();
12     }
13
14     public static int getDonutCount() {
15         return donutCount;
16     }
17
18     public static void main(String[] args){
19         Donut d1 = DonutFactory.cookDonut();
20         Donut d2 = DonutFactory.cookDonut();
21         System.out.println("count = " +
22             DonutFactory.getDonutCount());
23     }
24 }

```

Raison(s) de ou des erreur(s) :

▷ **Question 5.** (2 pt)

Considérer la classe suivante et indiquer **la/les réponses correcte(s)** si l'instruction proposée est insérée en ligne 25. Si il n'y a pas d'erreur, préciser l'affichage.

TestDynamicBinding.java

```

1 class Bidule {
2     void bipbip(Bidule x) {
3         System.out.println("bipbip de Bidule");
4     }
5     void coincoin(Bidule x) {
6         System.out.println("coincoin de Bidule");
7     }
8 }
9
10 class Machin extends Bidule {
11     void bipbip(Bidule x) {
12         System.out.println("bipbip de Machin");
13     }
14     void coincoin(Machin x) {
15         System.out.println("coincoin de Machin");
16     }
17 }
18
19 class Test {
20     public static void main(String[] argv) {
21         Bidule x = new Machin();
22         Machin y = new Machin();
23         Bidule z = new Bidule();
24
25         // replace here
26     }
27 }

```

	Erreur (compilation)	Erreur (exécution)	Ok	Affichage
x.bipbip(y);	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
y.bipbip(y);	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
x.bipbip(z);	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
z.bipbip(y);	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
x.coincoin(z);	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
x.coincoin(y);	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
y.coincoin(y);	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
z.coincoin(y);	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

▷ **Question 6.** (1 pt)

Indiquez les concepts mis en œuvre dans le sujet de TP portant sur la course (Race, Team, Racer, etc.)

Vrai	Faux	
<input type="checkbox"/>	<input type="checkbox"/>	l'encapsulation.
<input type="checkbox"/>	<input type="checkbox"/>	l'héritage.
<input type="checkbox"/>	<input type="checkbox"/>	le polymorphisme.
<input type="checkbox"/>	<input type="checkbox"/>	la délégation.
<input type="checkbox"/>	<input type="checkbox"/>	la généricité.

▷ **Question 7.** (2 pt)

Le programme suivant est constitué d'une seule classe. Réécrivez ce programme en utilisant l'héritage, le polymorphisme et la liaison dynamique.

Mess.java

```

1 import java.util.List;
2 import java.util.ArrayList;
3
4 class FormeGeometrique {
5     public static final int NON_DEFINI = 0;
6     public static final int RECTANGLE = 1;
7     public static final int CERCLE = 2;
8     public static final int TRIANGLE = 3;
9     public static final int COMPOSEE = 4;
10
11     private int type = NON_DEFINI;
12
13     private double a;
14     private double b;
15     private double c;
16
17     private List<FormeGeometrique> formes;
18
19     public FormeGeometrique() {
20         this.type = COMPOSEE;
21         this.formes = new ArrayList<FormeGeometrique>();
22     }
23
24     public FormeGeometrique(double r) {
25         this.type = CERCLE;
26         this.a = r;
27     }
28
29     public FormeGeometrique(double l, double ll) {
30         this.type = RECTANGLE;
31         this.a = l;
32         this.b = ll;
33     }
34
35     public FormeGeometrique(double c1, double c2, double c3) {
36         this.type = TRIANGLE;
37         this.a = c1;
38         this.b = c2;
39         this.c = c3;
40     }
41
42     public void add(FormeGeometrique f) {
43         this.formes.add(f);
44     }
45
46     public double calculerPerimetre() {
47         switch (this.type) {
48             case RECTANGLE:
49                 return 2.*a+2.*b;
50             case CERCLE:
51                 return 2.*Math.PI*a;
52             case TRIANGLE:
53                 return a+b+c;
54             case COMPOSEE:
55                 double s = 0.;
56                 for (FormeGeometrique f : formes)
57                     s += f.calculerPerimetre();
58                 return s;
59             default:
60                 return 0.;
61         }
62     }
63
64     public double calculerAire() {
65         switch (this.type) {
66             case RECTANGLE:
67                 return a*b;
68             case CERCLE:
69                 return Math.PI*a*a;
70             case TRIANGLE:
71                 return 1./4.*Math.sqrt((a+b+c)*(-a+b+c)*(a-b+c)*(a+b-c));
72             case COMPOSEE:
73                 double s = 0.;
74                 for (FormeGeometrique f : formes)
75                     s += f.calculerAire();
76                 return s;
77             default:
78                 return 0.;
79         }
80     }
81 }

```

```
82 |
83 | class DirtyProgram {
84 |     public static void main(String args[]) {
85 |         FormeGeometrique f1 = new FormeGeometrique(10,10);
86 |         FormeGeometrique f2 = new FormeGeometrique(10,15,17);
87 |         FormeGeometrique f3 = new FormeGeometrique();
88 |         f3.add(f1);
89 |         f3.add(f2);
90 |         System.out.println("perimetre = "+f3.calculerPerimetre());
91 |     }
92 | }
```

▷ **Question 8.** (1 pt)

Quel est l'intérêt de réaliser une telle transformation. Justifiez votre réponse.

▷ **Question 9.** (1 pt)

Quelle est la différence conceptuelle entre une interface et une classe abstraite ?

▷ **Question 10.** (1 pt)

Pourquoi Java autorise-t-il la réalisation multiple d'interface et non pas l'héritage multiple ?

▷ **Question 11.** (3 pt)

Considérer la classe suivante et indiquer **la réponse correcte** si l'instructions proposée est insérée en ligne 11.

```

1  class Machin {}
2
3  class Bidule {}
4
5  class Truc extends Machin {}
6
7  class Chose {
8      public static void main(String args[]) {
9          Object o = new Truc();
10         Machin a = new Machin();
11         // instruction a inserer
12     }
13 }

```

```

Truc t = (Truc) o;
Machin m = (Machin) o;
Bidule b = (Bidule) o;
Truc tr = (Truc) a;
Machin m2 = (Machin) a;
Bidule b2 = (Bidule) a;

```

Erreur (compilation)	Erreur (exécution)	Ok
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>