

NOM :

PRÉNOM :

▷ **Question 1.** (2 pt) Indiquer si l'affirmation est correcte ou non :

Vrai	Faux	
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Une classe abstraite (déclarée <b>abstract</b> ) ne peut pas être instanciée.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Le receveur d'un appel de méthode peut être de type primitif.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Un champ <b>static</b> indique que le champ est commun à toutes les instances de la classe.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Une méthode définie <b>private</b> est accessible dans toutes les classes appartenant au même package.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Le mot clé <b>extends</b> est utilisé pour spécifier une relation d'héritage.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Une sous-classe d'une classe concrète (non abstraite) peut être déclarée abstraite.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	La classe <b>Object</b> est une sous-classe de toutes les classes que l'on peut définir.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Une classe dont tous les champs sont déclarés <b>private</b> ne peut pas être déclarée <b>public</b> .

▷ **Question 2.** (1 pt)

Considérer les classes suivantes et indiquer **la ou les réponses correctes** :

```

1 package fr.esial;
2
3 public class TestVisibility {
4     public int j;
5     protected int k;
6     private int l;
7 }
    
```

Vrai	Faux	
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Le champ j est accessible dans toutes les classes de tous packages.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Le champ k est accessible dans toutes les classes de tous packages.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Le champ l est accessible dans toutes les classes de tous packages.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Le champ j est accessible par les sous-classes de TestVisibility.
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Le champ k est accessible par les sous-classes de TestVisibility.
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Le champ l est accessible par les sous-classes de TestVisibility.

▷ **Question 3.** (1 pt)

Considérer les classes suivantes et indiquer **la réponse correcte** :

```

1 class Shape {
2     private String color;
3
4     public Shape(String color) {
5         System.out.print("Shape");
6         this.color = color;
7     }
8 }
9
10 class Rectangle extends Shape {
11     public Rectangle() {
12         System.out.print("Rectangle");
13     }
14 }
15
16 public class TestConstructor {
17     public static void main(String[] args) {
18         new Rectangle();
19     }
20 }
    
```

- Ce code ne compile pas (erreur ligne 4).
- Ce code ne compile pas (erreur ligne 11).
- Ce code compile et le programme affiche : Shape
- Ce code compile et le programme affiche : Rectangle
- Ce code compile et le programme affiche : ShapeRectangle
- Ce code compile et le programme affiche : RectangleShape

▷ **Question 4.** (1 pt)

Considérer les classes suivantes et indiquer **la réponse correcte** :

```

1 class Parent {
2     public Parent() {
3         System.out.print("A");
4     }
5 }
6 class Child extends Parent {
7     public Child(int x) {
8         System.out.print("B");
9     }
10    public Child() {
11        this(123);
12        System.out.print("C");
13    }
14 }
15 public class TestConstructor2 {
16     public static void main(String[] args) {
17         new Child();
18     }
19 }
    
```

- Ce code ne compile pas (erreur ligne 7).
- Ce code ne compile pas (erreur ligne 11).
- Ce code compile et le programme affiche : ACB
- Ce code compile et le programme affiche : ABC
- Ce code compile et le programme affiche : BC
- Ce code compile et le programme affiche : AC

▷ **Question 5.** (1 pt)

Considérer les classes suivantes et indiquer la **réponse correcte** :

```

1 public class WhatAMess {
2     public static void main(String[] args) {
3         System.out.print("1");
4         try {
5             System.out.print("2");
6             if (true) throw new Exception();
7             System.out.print("3");
8         } catch (Exception e) {
9             try {
10                System.out.print("4");
11                if (true) throw new Exception();
12                System.out.print("5");
13            } catch (Exception ex) {
14                try {
15                    System.out.print("6");
16                    if (false) throw new Exception();
17                    System.out.print("7");
18                } catch (Exception ex2) {
19                    System.out.print("8");
20                } finally {
21                    System.out.print("9");
22                }
23            } finally {
24                System.out.print("A");
25            }
26            System.out.print("B");
27        }
28    }
29 }
    
```

- Ce code compile et le programme affiche : 1
- Ce code compile et le programme affiche : 123
- Ce code compile et le programme affiche : 12467B
- Ce code compile et le programme affiche : 124679AB

▷ **Question 6.** (2 pt)

Considérer la classe suivante et indiquer **la/les réponses correcte(s)** si les instructions proposées sont insérées en ligne 19. Si il n'y a pas d'erreur, préciser l'affichage.

```

1 class Animal {
2     public void eat() {
3         System.out.println("Generic animal eating");
4     }
5 }
6
7 class Horse extends Animal {
8     public void eat() {
9         System.out.println("Horse eating hay");
10    }
11
12    public void eat(String meal) {
13        System.out.println("Horse eating "+meal);
14    }
15 }
16
17 class Farm {
18     public static void main(String[] args) {
19         // insert instructions here
20     }
21 }
    
```

	Erreur (compilation)	Erreur (exécution)	Ok	Affichage
Animal a = new Animal(); a.eat();	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Generic animal eating
Horse h = new Horse(); h.eat();	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Horse eating hay
Animal ah = new Horse(); ah.eat();	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Horse eating hay
Horse ha = new Animal(); ha.eat();	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	incompatible types, found : Animal, required : Horse
Horse he = new Horse(); he.eat("apples");	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Horse eating apples
Animal a2 = new Animal(); a2.eat("treats");	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	eat() in Animal cannot be applied to (java.lang.String)
Animal ah2 = new Horse(); ah2.eat("carrots");	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	eat() in Animal cannot be applied to (java.lang.String)
Animal ah3 = new Horse(); ((Horse) ah3).eat("cabbage");	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Horse eating cabbage

▷ **Question 7.** (1 pt)

Considérer les classes suivantes et indiquer **la réponse correcte** :

```

1 class Car {
2     public static int velocity = 100;
3 }
4
5 public class TestDrive {
6     public static void accelerate(Car c) {
7         c.velocity += 30;
8     }
9
10    public static void main(String[] args) {
11        Car lamborghini = new Car();
12        accelerate(lamborghini);
13
14        Car gt500 = new Car();
15        accelerate(gt500);
16
17        System.out.println("speed="+gt500.velocity);
18    }
19 }
    
```

- Ce code compile et le programme affiche : speed=100
- Ce code compile et le programme affiche : speed=130
- Ce code compile et le programme affiche : speed=160

▷ **Question 8.** (2 pt)

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

```

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 checked="" type="checkbox"/>	bipbip de Machin
y.bipbip(y) ;	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	bipbip de Machin
x.bipbip(z) ;	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	bipbip de Machin
z.bipbip(y) ;	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	bipbip de Bidule
x.coincoin(z) ;	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	coincoin de Bidule
x.coincoin(y) ;	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	coincoin de Bidule
y.coincoin(y) ;	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	coincoin de Machin
z.coincoin(y) ;	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	coincoin de Bidule

## ▷ Question 9. (6 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 pizza.Main aux points identifiés dans le code source (indiqués par le marqueur // POINT\_? memory schema at this point).

Vous préciserez l'affichage obtenu sur la sortie standard pour l'ensemble de l'exécution.

```

1 package pizza;
2
3 public class Ingredient {
4     private String name;
5     private int quantity;
6
7     public Ingredient(String name, int quantity) {
8         this.name = name;
9         this.quantity = quantity;
10    }
11    public String getName() {
12        return this.name;
13    }
14    public void setQuantity(int quantity) {
15        this.quantity = quantity;
16    }
17    public int getQuantity() {
18        return this.quantity;
19    }
20    public Ingredient duplicate() {
21        return new Ingredient(this.name, this.quantity);
22    }
23    public boolean equals(Object o) {
24        if (o != null && o instanceof Ingredient) {
25            Ingredient ing = (Ingredient) o;
26            return (this.name.equals(ing.name)
27                && this.quantity == ing.quantity);
28        } else {
29            return false;
30        }
31    }
32    public String toString() {
33        return this.name+" "+this.quantity+" unit(s)";
34    }
35 }

```

```

1 package pizza;
2
3 public class Main {
4     public static void main(String[] args) {
5         Pizza mg = new Pizza("Marguarita");
6         Ingredient tom = new Ingredient("Tomatoes", 150);
7         mg.addIngredient(tom);
8         Ingredient mozz = new Ingredient("Mozarella", 100);
9         mg.addIngredient(mozz);
10
11        Pizza mg2 = new Pizza("Marguarita");
12        mg2.addIngredient(new Ingredient("Mozarella", 100));
13        mg2.addIngredient(new Ingredient("Tomatoes", 150));
14
15        System.out.println("POINT 1");
16        System.out.println(mg);
17        System.out.println(mg2);
18        System.out.println(mg2 == mg);
19        System.out.println(mg2.equals(mg));
20        // POINT_1 -- draw memory schema at this point
21
22        Pizza nap = mg.duplicate();
23        Ingredient ingFromMG = mg.getIngredient(0);
24        Ingredient ingFromNAP = nap.getIngredient(0);
25
26        System.out.println("POINT 2");
27        System.out.println(nap);
28        System.out.println(mg);
29        System.out.println(nap == mg);
30        System.out.println(nap.equals(mg));
31        System.out.println(ingFromMG == ingFromNAP);
32        System.out.println(ingFromMG.equals(ingFromNAP));
33        // POINT_2 -- draw memory schema at this point
34
35        Ingredient ingFromMG2 = mg2.getIngredient(1);
36        ingFromMG.setQuantity(ingFromMG.getQuantity()+10);
37        ingFromMG2.setQuantity(ingFromMG2.getQuantity()+10);
38        ingFromNAP.setQuantity(ingFromNAP.getQuantity()+10);
39        System.out.println("POINT 3");
40        System.out.println(mg);
41        System.out.println(mg2);
42        System.out.println(nap);
43        // POINT_3 -- draw memory schema at this point
44    }
45 }

```

```

1 package pizza;
2
3 import java.util.List;
4 import java.util.ArrayList;
5 import java.util.Iterator;
6
7 public class Pizza {
8     private String name;
9     private List<Ingredient> ingredients;
10
11    public Pizza(String name) {
12        this.name = name;
13        this.ingredients = new ArrayList<Ingredient>();
14    }
15    public void setName(String name) {
16        this.name = name;
17    }
18    public String getName() {
19        return this.name;
20    }
21    public void addIngredient(Ingredient ing) {
22        this.ingredients.add(ing);
23    }
24    public Ingredient getIngredient(int index) {
25        return this.ingredients.get(index);
26    }
27    public Pizza duplicate() {
28        Pizza d = new Pizza(this.getName());
29        Iterator<Ingredient> it = this.ingredients.iterator();
30        while (it.hasNext())
31            d.addIngredient(it.next());
32        return d;
33    }
34    public boolean equals(Object o) {
35        if (o == null || !(o instanceof Pizza))
36            return false;
37        Pizza p = (Pizza) o;
38        if (!this.name.equals(p.name))
39            return false;
40        if (this.ingredients.size() != p.ingredients.size())
41            return false;
42        Iterator<Ingredient> it = p.ingredients.iterator();
43        while (it.hasNext()) {
44            if (!this.ingredients.contains(it.next()))
45                return false;
46        }
47        return true;
48    }
49    public String toString() {
50        String s = this.name+"[";
51        for (int i=0; i<this.ingredients.size(); i++) {
52            if (i > 0)
53                s += ", ";
54            s += this.ingredients.get(i);
55        }
56        s += "]";
57        return s;
58    }
59 }

```

Réponse

POINT 1

Margarita[Tomatoes 150 unit(s), Mozarella 100 unit(s)]

Margarita[Mozarella 100 unit(s), Tomatoes 150 unit(s)]

false

```
true
POINT 2
Marguarita[Tomatoes 150 unit(s), Mozarella 100 unit(s)]
Marguarita[Tomatoes 150 unit(s), Mozarella 100 unit(s)]
false
true
true
true
POINT 3
Marguarita[Tomatoes 170 unit(s), Mozarella 100 unit(s)]
Marguarita[Mozarella 100 unit(s), Tomatoes 160 unit(s)]
Marguarita[Tomatoes 170 unit(s), Mozarella 100 unit(s)]
```

---

Fin réponse

▷ **Question 10.** (1 pt)

Écrire le code de la méthode `mostUsedIngredient()` de la classe `pizza.Pizza` qui retourne une référence vers l'ingrédient d'une pizza dont la quantité est la plus élevée. Dans le cas où il y aurait plusieurs `Ingredient` avec la même quantité, le dernier `Ingredient` trouvé dont la quantité est la plus élevée sera retourné.

```
1 public Ingredient mostUsedIngredient() {
2     int quantity = 0;
3     Ingredient res = null;
4     for (int i = 0; i < this.ingredients.size(); i++) {
5         Ingredient ing = this.ingredients.get(i);
6         if (ing.getQuantity() >= quantity) {
7             res = ing;
8             quantity = res.getQuantity();
9         }
10    }
11    return res;
12 }
```

---

Fin réponse