

LASTNAME :

FIRSTNAME :

▷ **Question 1.** (2 pt) For each sentence, please indicate if it is right or wrong :

True	False	
<input type="checkbox"/>	<input type="checkbox"/>	An abstract class (marked as <b>abstract</b> ) cannot be instantiated.
<input type="checkbox"/>	<input type="checkbox"/>	The receiver of a method call can be of primitive type.
<input type="checkbox"/>	<input type="checkbox"/>	A <b>static</b> field is a field which is common to all instances of a class.
<input type="checkbox"/>	<input type="checkbox"/>	A method defined to be <b>private</b> can be called from all classes belonging to the same package.
<input type="checkbox"/>	<input type="checkbox"/>	The keyword <b>extends</b> is used to declare an inheritance relationship.
<input type="checkbox"/>	<input type="checkbox"/>	A sub-class from a concrete class (not an abstract one) can be declared abstract.
<input type="checkbox"/>	<input type="checkbox"/>	The <b>Object</b> class is a sub-class of all classes that can be defined.
<input type="checkbox"/>	<input type="checkbox"/>	A class with all fields declared <b>private</b> cannot be declared as <b>public</b> .

▷ **Question 2.** (1 pt)

Consider the following classes and indicate the **correct answer** (there might be several correct answers) :

```

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

```

True	False	
<input type="checkbox"/>	<input type="checkbox"/>	The field <b>j</b> can be accessed from all classes of all packages.
<input type="checkbox"/>	<input type="checkbox"/>	The field <b>k</b> can be accessed from all classes of all packages.
<input type="checkbox"/>	<input type="checkbox"/>	The field <b>l</b> can be accessed from all classes of all packages.
<input type="checkbox"/>	<input type="checkbox"/>	The field <b>j</b> can be accessed from all sub-classes of <b>TestVisibility</b> .
<input type="checkbox"/>	<input type="checkbox"/>	The field <b>k</b> can be accessed from all sub-classes of <b>TestVisibility</b> .
<input type="checkbox"/>	<input type="checkbox"/>	The field <b>l</b> can be accessed from all sub-classes of <b>TestVisibility</b> .

▷ **Question 3.** (1 pt)

Consider the following classes and indicate the **correct answer** :

```

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 }

```

- ☐ This code does not compile (error at line 4).
- ☐ This code does not compile (error at line 11).
- ☐ This code compiles and the program output is : **Shape**
- ☐ This code compiles and the program output is : **Rectangle**
- ☐ This code compiles and the program output is : **ShapeRectangle**
- ☐ This code compiles and the program output is : **RectangleShape**

▷ **Question 4.** (1 pt)

Consider the following classes and indicate the **correct answer** :

```

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 }

```

- ☐ This code does not compile (error at line 7).
- ☐ This code does not compile (error at line 11).
- ☐ This code compiles and the program output is : **ACB**
- ☐ This code compiles and the program output is : **ABC**
- ☐ This code compiles and the program output is : **BC**
- ☐ This code compiles and the program output is : **AC**

## ▷ Question 5. (1 pt)

Consider the following classes and indicate the **correct answer** :

```

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        }
27    }
28 }
29 }

```

- ☐ This code compiles and the program output is : 1  
☐ This code compiles and the program output is : 123  
☐ This code compiles and the program output is : 12467B  
☐ This code compiles and the program output is : 124679AB

## ▷ Question 6. (2 pt)

Consider the following classes and indicate the **correct answer** (there might be several correct answers) when the provided instructions are inserted at line 19. If there is no error then please indicate the program output.

```

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 }

```

	Error (compilation)	Error (execution)	Ok	Output
Animal a = new Animal(); a.eat();	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Horse h = new Horse(); h.eat();	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Animal ah = new Horse(); ah.eat();	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Horse ha = new Animal(); ha.eat();	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Horse he = new Horse(); he.eat("apples");	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Animal a2 = new Animal(); a2.eat("treats");	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Animal ah2 = new Horse(); ah2.eat("carrots");	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Animal ah3 = new Horse(); ((Horse) ah3).eat("cabbage");	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

▷ **Question 7.** (1 pt)

Consider the following classes and indicate the **correct answer** :

```

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 }

```

- ☐ This code compiles and the program output is : speed=100
- ☐ This code compiles and the program output is : speed=130
- ☐ This code compiles and the program output is : speed=160

▷ **Question 8.** (2 pt)

Consider the following classes and indicate the **correct answer** (there might be several correct answers) when the provided instructions are inserted at line 25. If there is no error then please indicate the program output.

```

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 }

```

	Error (compilation)	Error (execution)	Ok	Output
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 9. (6 pt)

Consider the following classes and at each step (indicated by the marker // POINT\_? memory schema at this point) draw a schema representing the memory state (stack and heap) during the execution of the method main() of the pizza.Main class.

Please indicate the program output (standard output) at the end of the program execution.

```

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 }

```

## ▷ Question 10. (1 pt)

Write the code of the mostUsedIngredient() method from the pizza.Pizza class. This method will return an object reference to the pizza's ingredient whose quantity is the highest. In case that several Ingredient are used in the same quantity, the last found Ingredient whose quantity is the highest will be returned.