

INTRODUCTION TO PROGRAMMING

Test Semester I

Instructions :

- The exam lasts one hour and 45 minutes (8h15 - 10h);
- Maximum number of points is 125 points (from which approximately 30 are bonus);
- All documentation is permitted;
- Be sure to do **only one exercise per sheet**, and **indicate your sciper number** on *every* sheet.
A sheet without an identification is not going to be graded;
- The exam consists of 4 exercises. Those exercises are independent.
- The exercises are not of the same difficulty. Start with the exercises that use the concepts you are the most familiar with.

QUESTIONS ON THE NEXT PAGE

Exercise 1 : Concepts [33 points]

Answer clearly and briefly the following questions :

1. [10 points] Consider the following code :

```
1. class X
2. {
3.     public X m() {
4.         return new X(this);
5.     }
6.     public X() {}
7.     public X(X o) {}
8. }

9. class Y extends X
10. {
11.     public Y m() {
12.         return new Y(this);
13.     }

14.     public Y() {}

15.     public Y(Y o) {super(o);}
16. }


17. class Test
18. {
19.     public static void main(String[] args) {
20.         Y y1 = new Y();
21.         Y y2 = new Y(y1);
22.         System.out.println( y1.equals(y2));
23.     }
24. }
```

- (a) What does the program print? Briefly explain.
(b) Would it still compile if we remove the instruction `super(o);` from line 15? Briefly explain.
(c) Is the method `m` from `Y` a redefinition (override) of the method `m` from `X` ? Explain.
(d) Would the code still compile if we add the following method to the class `Y` :

```
public Object m() {
    return new Y(this);
}
```

Briefly explain.

- (e) Can we rely on a default constructor if we remove the copy constructor from the class `Y` ?
Briefly explain.

Continued on the reverse 

2. [3 points] Consider the following program :

```
1. class X
2. {
3.     private String x = "X";
4. }

5. class Y extends X
6. {
7.     public Y() {
8.         super("X");
9.     }

10.    public Y(String s) {
11.        this(s);
12.    }
13. }
```

Indicate the error(s) preventing this program to compile.

3. [4 points] Consider the following code snippet :

```
class A {
    private int a1;
    private int a2;
    private int a3;

    public A(int val1, int val2, int val3) {
        a1 = val1;
        a2 = val2;
        a3 = val3;
    }
}
```

How do you propose to complete the code, without duplicating lines of code, to enable the construction of instances of A as follows :


```
A a1 = new A(1,2); // the attribute a3 takes 10 as a default value
A a2 = new A(1);   // the attributes a2 and a3 take 10 as default values
```

-
4. [3 points] Consider the following code :

```
interface Dessinable
{
    void draw();
}

class Ballon implements Dessinable
{
    void draw() {
        // on dessine le ballon ici
    }
}
```

- (a) Why does the code not compile?
- (b) How do you propose to fix it?
5. [3 points] Does a static attribute from a class have to be initialized by one of the constructors of the class? Briefly explain your answer.
6. [3 points] Why are the `RuntimeException`(s) not subject to the rules «declare-or-treat» («déclarer-ou-traiter»)?

Continued on the reverse 

7. [7 points] Consider the following code :

```
class Disque
{
    private int rayon;

    public Disque(int unRayon) {
        rayon = unRayon;
    }

    boolean equals(Disque o) {
        return rayon == o.rayon;
    }
}

class equals
{
    public static void main(String[] args) {
        Disque d1 = new Disque(3);
        Disque d2 = new Disque(1+2);
        System.out.println(d1.equals(d2));
        System.out.println(d1.equals("Un joli disque de rayon 3"));
    }
}
```

- (a) Why can the method `equals` from the class `Disque` access the radius (`rayon`) of the object `o` without using a public getter?
- (b) Does the method `main` compile?
 - i. if yes, explain why and what does it print.
 - ii. if not, explain why and how one can fix it.

CONTINUED ON THE NEXT PAGE

Exercise 2 : OO concepts and programming [50 points + 15 bonus]

It is important to read the entire description before starting to respond to the questions.

In this exercise we simulate the evolution of bacteria in a substrate.

The substrate A substrate is characterized by a *temperature* that is supposed to be constant (when a value is assigned that value will not change thereafter). It contains a *group of bacteria* and a *group of food sources for these bacteria*. It has a circular shape (a Petri dish) characterized by a *radius*, also constant.

The required features of the substrate are :

- being able to construct a substrate by assigning a temperature and a radius ;
- being able to add in it a given bacterium ;
- being able to add in it a given source of food ;
- being able to make the bacteria population evolve over time.

The sources of food A food source has a *position* on the substrate and is characterized by a *quantity* (an integer for simplicity).

The bacteria Each bacterium :

- has a *position* on the substrate,
- has an *energy level* (an integer) and a movement speed;
- can move, divide itself and die;
- evolves over time;

You will suppose that there are predefined classes `Position` and `Speed` that can be used to model the position and velocity.

A bacterium dies when its energy level reaches zero.

Two types of bacteria are grown: the *bacteria flagella* and *tentacle bacteria*. A tentacle bacterium is characterized by the length of its tentacle.

The two types of bacteria distinguish from each other by the movement mode : the first move at a constant speed by changing direction at random, the second move by means of a tentacle.

The exact implementations of the moving alternatives and division do not concern us here.

The algorithm that simulates the evolution of bacteria is the same for all types of bacteria. It implements the following steps at each simulation phase :

1. the bacterium moves. If it reaches the borders of the substrate it reverses direction;
2. if a food source is close enough, the bacterium consumes it and increases its energy level by the amount of food consumed. The bacterium consumes a fixed amount of food each time and this amount is the characteristic of the bacteria (each bacterium will have a fixed amount fixed at birth) ;
3. the bacterium checks its energy level and decides to divide itself according to that level;
4. the bacterium loses a certain amount of energy (fixed and identical for all types of bacteria) ;

The division depends on the substrate temperature and requires different energy threshold levels for different types of bacteria. The nature of the temperature influence on the division does not interest us.

The fully consumed food sources and dead bacteria must disappear from the substrate.

Assume :

- that the method making the bacteria evolve over time has the following signature `evolve(Substrate s)` where `Substrate` is the class modeling the substrate;
- that the substrate **provide no method giving access to its whole sets of food sources or bacteria** (Hint : to consume food for example, a bacteria only need to find the food source which is the closest to its own position);
- the movement methods are specific to a category of bacteria and can not be defined concretely for one bacterium;

Part 1 : Design (43 points)

Draw a hierarchy of classes needed to implement the desired functionality. In your diagram, specify classes, potential interfaces, attributes and the headers of methods (**without body**).

Design Constraints

1. Your design should use polymorphism. It should not need to duplicate the code, have unnecessary getters/setters nor type tests.
2. The constructors should not be forgotten. Assume they initialize the attributes using values passed as parameters.
3. Pay a special attention to the access rights of the attributes and methods and indicate clearly what is abstract, final and static in your design.

Part 2 : Programming (7 points + 3 bonus)

Write a code describing the evolution of a bacterium. (**Bonus** : what should we modify in the design if `Substrate` was not a parameter of the `evolve` method of bacteria?)

Part 3 : Design (bonus, 12 points)


Assume now that you want to simulate the interactions between bacteria. A bacterium can therefore meet another and potentially kill it when meeting. Assume that :

- bacteria flagella can not kill any bacteria;
- the tentacle bacteria can only kill bacteria flagella.

Explain how you would proceed to implement the method :

`void interact(Bacterium other)` (adapt the name of the class if you named it otherwise) to manage the meeting of a bacterium with another without making any type tests.

Provide the necessary Java code and indicate in which class(es) to place it.

Continued on the reverse 
--

Exercise 3 : Program analysis [20 points]

A software development company asks you to evaluate its program for resource management (human and material). Employees are categorized schematically in this program:

```
abstract class Employee
{
    // plein de choses
    public abstract double salaire();
}

class Consultant extends Employee
{
    // plein de choses
    public double salaire();
}

class Commercial extends Employee
{
    // plein de choses
    public double salaire();
}

class Technicien extends Employee
{
    // plein de choses
    public double salaire();
}

class Developpeur extends Employee
{
    // plein de choses
    public double salaire();
}
```

3.1 Bonus payment (6 points)

The director of the human resources demands you to add a feature to assign an end-of-year bonus to all permanent employees of the company :

```
void payerBonus(Employee[] desEmployes, double bonus)
```

(from the list `desEmployes`, only the permanent employees should receive a bonus)

Permanent employees are the `Developpeur`(s) and the `Technicien`(s) (but no information in the current hierarchy allows to know that).

How would you proceed to code such a functionality **without doing a type check** ? You have the right to enrich the existing class hierarchy.

3.2 Employee shares (7 points)

By observing the constructor of the main class (which allows to recruit a set of employees at once) :

```
class Entreprise
{
    private Employee[] staff;

    public Entreprise(Employee[] unStaff) {
        staff = unStaff;
    }

    //plein de choses
}
```

you find privacy leak(s)(«faible d'encapsulation») (note that the employee instances are mutable: an employee can change its wage category, place of residence etc.).

Explain the flaw and how to fix it (it is not necessary to give the code. Explain just the principle). Is it eventually necessary to make changes to the content of the class hierarchy to fix this flaw?

Continued on the reverse ➡

3.3 Sources of expenses (7 points)

The management program also includes the following components :

```
// le matériel acheté et utilisé par l'entreprise
abstract class Materiel
{
    public abstract double prixAchat();
}

class Ordinateur extends Materiel
{
    public double prixAchat() { ... }
}

class Licence extends Materiel
{
    public double prixAchat() { ... }
}

// les logiciels fabriqués et vendus par l'entreprise
class Logiciel
{
    public double prixVente();
}
```

In the program there exists a functionality :

```
double estimationGain(Logiciel[] s, Employee[] e, Materiel[] m)
```

that estimates the gains of the company when selling software **s**, knowing that for its development it is necessary to pay the salary of employees **e** and buy the material **m** (essentially, $\text{gain} = \text{price gained by selling the software} - \text{the employees salary} - \text{cost of the material needed for the development}$).

The company's director wants this feature not to explicitly expose its sources of expenses and to allow to add other expenses without requiring subsequent changes.

Which solution would you propose?

CONTINUED ON THE NEXT PAGE

Exercise 4 : Exceptions [22 points]

Consider the following program :

```
1. import java.util.Scanner;
2. import java.util.ArrayList;

3. class MyReader
4. {
5.     private final static Scanner clavier = new Scanner(System.in);
6.     public static int readInt() {
7.         int lu;
8.         System.out.println("Donnez un entier");
9.         lu = clavier.nextInt();
10.        return lu;
11.    }
12. }

13. class Container
14. {
15.     private ArrayList<Integer> collection;

16.     public Container() {
17.         collection = new ArrayList<Integer>();
18.     }

19.     public void read(int aSize) {
20.         for (int i = 0; i < aSize; ++i) {
21.             collection.add(1/MyReader.readInt());
22.         }
23.     }
24. }

25. class Exceptions
26. {
27.     public static void main(String[] args) {
28.         Container myContainer = new Container();
29.         myContainer.read(20);
30.     }
31. }
```

-
1. How will the program behave if the user enters :
 - (a) a value other than integer ?
 - (b) a zero ?
 2. How do you propose to change the code, by using exception handling, to enable the user to :
 - (a) have right to make an error 10 times in case a value other than integer is entered. After 10 times, the program should launch an `InputMismatchException`, which will be addressed by the main program.
 - (b) have 5 times right to make an error in case 0 is entered as an array entry ((s)he has the right to repeat 5 times in case 0 is entered as one entry of the array). After 5 times, the program should launch an `InputMismatchException`, which will be addressed by the main program.
Important : the `readInt`, must not consider the input of a zero as an error/exception. `readInt` is expected to be used in other contexts.

The main program will handle the thrown exceptions and allow the display of appropriate messages ("Too many zeros when filling the array !" or "You have entered a non integer value too many times !").

Indicate which line(s) of code to add and where.