ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communication
Cours Introduction à la Programmation, (SIN/SSC)
Sam J.

INTRODUCTION A LA PROGRAMMATION

Examen semestriel

Instructions :

Instructions :

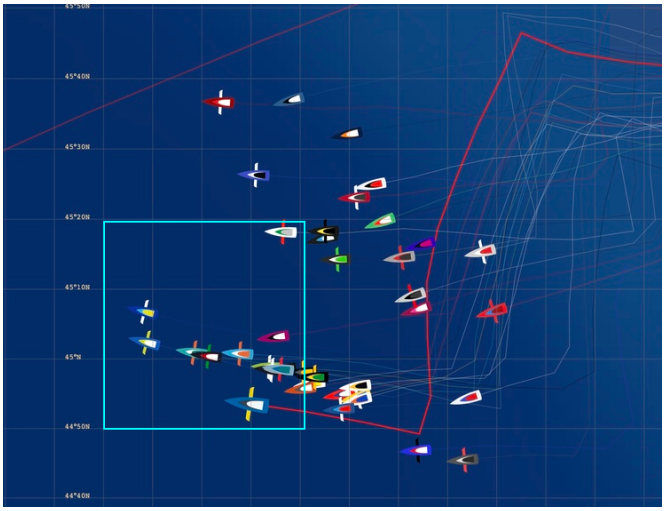
- You have three hours to do this exam (9h00 - 12h00).
- Maximum 105 points (15 points are considered as bonus).
- Attention : There are also statements on the back of the pages.
- You must **write in black or dark blue ink**, not with a pencil and not with any other color.
- Any paper documentation is allowed ; However, you may not use a personal computer, cell phone, or other electronic equipment.
- Answer on the sheets given to you and **in the places provided**. **Do not write your answers in this document**. Additionnal answer sheets are provided at the end of the answering booklet.
- Do not attach any additional sheets ; **only answer sheets that were distributed to you will be graded**.
- You can answer the questions in French or in English.
- The exam consists of 3 independent exercises. **You can start with whichever you want**
  - Exercice 1 : **50** points.
  - Exercice 2 : **38** points.
  - Exercice 3 : **17** points.

suite au verso ☞

Exercice 1 : Conception OO [50 points]

It is important to read each question of this exercise in its entirety before answering. Take the time to read the imposed constraints and consult the code provided in the appendix before starting the exercise. This exercise contains 3 questions.

The Vendée Globe is a famous solo sailing race. The organization has set up a visualization tool to track the skippers live. We propose to structure this tool using an object-oriented approach.



The visualization tool should typically allow the display of characteristic data of the boats or trajectory projections for them.

Design Constraints When solving this exercise, the following constraints must be respected :

1. When you are asked to write the body of a class, you will do so in Java syntax and include the attributes and methods but without their implementations (except for those explicitly requested).
2. You will assume that there is a class `API` available through an import directive, which allows retrieving information related to the boats. You will also assume that a main program exists, which creates the necessary objects, repeatedly calls an update method, as well as a draw method on a `Canvas` object.
3. The classes must contain the necessary members to implement all the desired functionalities, **whether they are explicitly requested or suggested by the specifications in the statement** ;
4. For methods that are not trivial, **you will include a brief comment explaining the meaning of the return type, what they do in broad terms, and the functionalities of other classes they use**.
5. **Do not neglect either the constructors or the access rights**.
6. The `protected` access modifier must not be used for attributes.
7. Your design must be well modularized.
8. You should only provide getters/setters when they are necessary for implementing the described simulation.
9. It is not necessary to write import directives.

suite ☞

Question 1 : Boats and map (28.5 points)

A *boat* is characterized by the *name of its skipper*, its *nautical coordinates* (a `NavigationCoord`), its *speed* (a `double`), its *rank* in the race, its *distance* from the lead boat, and an *identifier* (an integer unique to it). These data are initialized during construction using parameters. The rank, speed and distance are zero at the start.

Boats can be of different types, to simplify : *Monohulls* or *Catamarans*. The latter can be equipped with sails or not. A map, to be visualized by the race tracking tool, contains at this stage a *set of boats* and the nautical coordinate (`NavigationCoord`) on which its view is centered (this is the center of the map as visualized by the tool). These data are also provided during construction. If the center coordinate is not provided, it will have a default value corresponding to the constant provided by `API`.

The following basic functionalities are desired at this stage :

1. Be able to individually access all the characteristic information of the boats ;
2. change the center of the map ;
3. draw a map on a `Canvas` ; which implies drawing all its boats ;
4. update, using a single method, the characteristic data of the boats on the map using the tools of the `API` ;
5. remove a boat with a given identifier from the map.

It is assumed that the drawing of boats can only be concretely implemented for specific types of boats. We also want the update method to be able to use potential subclasses of `API` (which would be determined in the main program), and that the `API` should not be communicated to the boats. **Write the classes and any necessary interfaces to represent the model described above. You will also provide the complete implementation of the update methods**.

Question 2 : Marine data fields (16.5 points)

We would now like to complete the tool by visualizing estimated optimal trajectories for a given boat.

The estimation of a trajectory can be made by algorithms that we assume are known but specific to each type of boat. These algorithms depend on *maritime data fields* (`Field`, turquoise grid in the figure) that provide information on wind, ocean currents, etc. A `Field` is a square grid characterized by the nautical coordinate (`NavigationCoord`) of its top-right corner, and its size (an integer for simplicity). Each cell of its grid contains data of the provided type `Data`. Two types of `Field` will be considered : *wind fields* (with data related to the wind, provided type `WindData`) and `ocean current fields` (with data related to ocean currents, provided type `OceanCurrentData`).

The desired functionalities for `Field` are as follows :

1. Construct a field by assigning it coordinates, a size, and an empty grid ;
2. returns the nautical coordinate corresponding to the center of the cell at coordinate `[i][j]` of the grid ;
3. fill the grid with data, ensuring that it contains only data of type `WindData` for wind fields and data of type `OceanCurrentData` for ocean current fields. This filling will be done using the `getWindValue()` and `getOceanCurrentValue()` methods of the API. Additionally, it is desired that this filling functionality only accesses a restricted view of the API, offering only the methods `getWindData` and `getOceanCurrentData()`.

Write the classes and any necessary interfaces to represent the model described above. You will also provide the complete implementation of the update methods.

Question 3 : Optimal Trajectories (5 points)

The additional functionality we wish to have for the boats is the calculation of an estimated optimal trajectory (provided type `Trajectory`) based on a **variable** set, potentially empty, of `Field` types.

For example, we want to be able to calculate the optimal trajectory taking into account a `Field` of type *"wind"* and a `Field` of type *"ocean current"* (of the same size and position), or calculate the trajectory considering only a `Field` of type *"wind"* or only a `Field` of type *"ocean current"*.

The method calculating an estimated optimal trajectory will apply the algorithms specific to the types of boats (which use the field data).

Provide the method headers to be added to your design to enable the calculation of optimal trajectories, specifying where to place them.

You will consider that new field types can be added to the design later.

Exercice 2 : Concepts de base [38 points]

Answer the following questions clearly and succinctly :

1. **[6 points]** Consider the following code :

```
1  public class BitManipulation {
2
3      public static void main(String[] args) {
4        //12 = 00000000_00000000_00000000_00001100 en binaire
5        int value = 12;
6        System.out.println(getXthBit(value, 3));
7        System.out.println(getXthBit(value, value));
8
9        int newValue = embedInXthBit(value, false, 2);
10       print(value, newValue);
11       // -12 = 11111111_11111111_11111111_11110100 en binaire
12       value = -12;
13       newValue = embedInXthBit(value, false, 2);
14       print(value, newValue);
15     }
16
17     public static void print(int value, int newValue){
18       System.out.println("Original Value: " + value);
19       System.out.println("Modified Value: " + newValue);
20     }
21
22     public static boolean getXthBit(int value, int pos) {
23       assert 0 <= pos && pos < Integer.SIZE;
24       final int lsb = value & (0x1 << pos);
25       return lsb != 0;
26      }
27
28     public static int embedInXthBit(int value, boolean m, int pos
          ) {
29       assert 0 <= pos && pos < Integer.SIZE;
30       final int mask = 0x1 << pos;
31       return m ? value | mask : value & ~mask;
32     }
33
34 }
```

(a) Briefly explain how the `getXthBit` method works.

(b) Briefly explain how the `embedInXthBith` method works.

(c) Indicate what it displays.

2. **[6 points]** Consider the following code :

```
1  class W {
2      private String w = "colugo";
3      public String toString(){ return w;}
4      public void setW(String w){ this.w = w; }
5  }
6  class Pgme {
7    public static void main(String[] args){
8      W w1 = new W(); W w2 = new W();
9      m(w1,w2);
10     System.out.println(w1);
11     System.out.println(w2);
12   }
13    public static void m(W w1, W w2){
14     w1 = w2;
15     w2.setW("hyrax");
16    }
17 }
```

(a) What does it display ?

(b) What is displayed if we insert the instruction `w1=w2;` between lines 8 and 9 ?

(c) What does it display if you delete `this.` on line 4 ?

Briefly justify each of your answers.

3. **[12.5 points]** Consider the following code :

```
1  class A {
2      private String a = "A";
3      public A() { System.out.println(a); }
4      public String toString(){ return a;}
5  }
6  class B extends A {
7     private String b = "B";
8     public B() { System.out.println(b); }
9     public B(String b) { this.b  = b; }
10    public String toString(){ return super.toString() + b; }
11 }
12 class C extends B {
13    private String c = "C";
14    public String toString(){ return super.toString() + c; }
15 }
16
17 class Ctor {
18     public static void main(String[] args){
19        C c = new C();
20        System.out.println(c);
21     }
22 }
```

(a) What does it display ?

(b) What happens if we replace `super` with `this` in line 10 ?

(c) Does the program still compile if you delete the `B` constructor of line 8 ?

(d) Does the program continue to run succesfully if you delete the `toString` method from `B` ?

(e) Is it possible to add `this();` as the first instruction of the default `B` constructor ?

(f) Is it possible to add `this("b")` as the first instruction of the default `B` constructor ?

(g) Is it possible to add `super();` as the first instruction in the default constructor of `B` ?

(h) Is it possible to add `super("b")` as the first instruction in the default constructor of `B` ?

Briefly justify each of your answers.

suite au verso ☞

4. **[13.5 points]** Consider the following program, using which you want to display a description of the items stored in a library, their number, and the price of each item.

```
1  abstract class ShelfItem {
2    private String title;
3    public ShelfItem (String title){ this.title = title;}
4    public abstract String getId();
5    public int getPages() {return -1; }
6    public String getTitle() {return title; }
7  }
8
9  class Book extends ShelfItem {
10   private final int nbPages;
11   public Book(String title, int nbPages) {
12   super(title); this.nbPages= nbPages;
13   }
14   public String getId() { return "book::" + getTitle(); }
15   public int getPages() { return nbPages; }
16 }
17
18 class DVD extends ShelfItem {
19   public DVD(String name) { super(name); }
20   public String getId() { return "dvd::" + getTitle(); }
21 }
22
23 class Shelf {
24   public static int CAPACITY = 200;
25   private ShelfItem[] items = new ShelfItem[CAPACITY];
26   private int numItems = 0;
27
28   public void add(ShelfItem item) {
29     if (numItems < CAPACITY-1)
30       items[numItems++] = item;
31   }
32
33   public String toString(){
34     String description = "";
35     for (int i=0; i < numItems; ++i){
36       description += items[i].getId();
37       int pages = items[i].getPages();
38       if (pages != -1) {
39         description += "," + Integer.toString(pages)
40                     + " pages";
41       }
42       description += "\n"; // new line
43     }
44     return description;
45   }
46   public ShelfItem[] getAllItems() {return items;}
47 }
48
```

```
49
50 class PriceCalculator {
51   public static float price(Shelf shelf) {
52     float price = 0.0f;
53     for (ShelfItem item : shelf.getAllItems())
54       price += price(item);
55       return price;
56   }
57
58   public static float price(ShelfItem item) { return 0.0f; }
59
60   public static float price (Book book) { return 5.0f; }
61
62   public static float price(DVD dvd) { return 2.0f; }
63 }
64
65
66 public class StaticPolym {
67   public static void main(String[] args) {
68     Shelf shelf = new Shelf();
69     Book book = new Book("The old man and the waves", 250);
70     shelf.add(book);
71     DVD dvd = new DVD("the-gleaming");
72     shelf.add(dvd);
73     System.out.println(" The content of the shelf is:\n" + shelf)
          ;
74     System.out.println("The Shelf has : "
75               + shelf.getAllItems().length + " items");
76     float totalPrice = PriceCalculator.price(shelf);
77     System.out.println("The shelf price is : " + totalPrice);
78   }
79 }
```

(a) What display does it produce ?

(b) What criticism can you make of the implementation of the `getPages()` method and what better solution would you suggest ?

(c) What critique can you make of the getter `getAllItems()`, and what better solution would you propose ?

(d) What other modifications would you propose to produce the initially desired display ?

An explanation in French or English is sufficient for questions (b) to (d), but you may instead provide the code you would propose to replace the provided one.

suite au verso ☞

Exercice 3 : Déroulement de programme [17 points]

The following program runs without errors (assume the import directives are present). What does it display ? Briefly explain its execution **without paraphrasing the code**.

```
1  import java.lang.reflect.Array;
2  import java.util.ArrayList;
3
4  interface IdGenerator {
5      int generateId();
6  }
7  interface DefaultIdGenerator extends IdGenerator {
8      int START_ID = 0;
9      default int generateId() { return START_ID; }
10 }
11 class SequentialIdGenerator implements DefaultIdGenerator {
12     private static final int START_ID = 5;
13     private static final SequentialIdGenerator instance
14                                 = new SequentialIdGenerator();
15     public static SequentialIdGenerator getInstance() {
16                                             return instance; }
17     private int nextId;
18
19     protected SequentialIdGenerator(int startId) {
20                                             nextId = startId; }
21     private SequentialIdGenerator() {
22         this(DefaultIdGenerator.START_ID + START_ID);
23     }
24     @Override
25     public int generateId() {
26         System.out.println("Generating ID: " + (++nextId));
27         return nextId++;
28     }
29 }
30 interface Saluter {
31     default String salute(InformalUser user){
32         return "Hello " + user;
33     }
34     default String salute(FormalUser user){
35         return "Delighted to see you " + user;
36     }
37 }
38 interface Salutable {
39     String acceptSalutation(Saluter saluter);
40 }
41
42 abstract class User implements Salutable {
43     private int id;
44     private String name;
45     private String username;
46     public User() { this("User"); }
47
```

```
48      public User(String name) {
49          this.name = name;
50          this.username = name.toLowerCase();
51          id = SequentialIdGenerator.getInstance().generateId();
52      }
53      @Override
54      public String toString() { return getUsername() + "#" + id; }
55      public String getUsername() { return username; }
56  }
57
58  class InformalUser extends User {
59      private String name;
60      public InformalUser(String name) {
61          super();
62          this.name = name;
63      }
64      @Override
65      public String acceptSalutation(Saluter saluter) {
66          return saluter.salute(this);
67      }
68      @Override
69      public String toString() {
70          return name + " (" + getUsername() + ")";
71      }
72  }
73  class FormalUser extends User {
74      public FormalUser(String name) { super(name); }
75      @Override
76      public String acceptSalutation(Saluter saluter) {
77          return saluter.salute(this);
78      }
79  }
80  class Platform implements Saluter {
81      private ArrayList<User> users = new ArrayList<>();
82      public void add(User user) { users.add(user); }
83      public void welcomeUsers() {
84          for (User user : users) {
85                System.out.println(user.acceptSalutation(this)); }
86      }
87  }
88
89  public class Deroulement {
90      public static void main(String[] args) {
91          Platform platform = new Platform();
92          platform.add(new InformalUser("Bob"));
93          platform.add(new FormalUser("Alice"));
94          platform.add(new FormalUser("Charlotte"));
95          platform.welcomeUsers();
96      }
97  }
```