Computer Science Exam
2nd year - Janvier 2017-2018

INSA

Duration : 2h
Authorised documents: *All personal notes and lecture notes.*

— The grading scale is over 24 marks and is given on an indicative basis.
— The exam subject comprises 6 pages.

We want to program a game which consists in getting a character (the hiker) out of a maze. A picture of this game is given in figure 1 (left hand side). The hiker is represented by the blue ellipse. He starts from the red square (the starting point, at the bottom left of the maze) and needs to get to the green square (the finish point, at the top right of the maze). The player moves the hiker by using buttons situated at the bottom of the screen. Maze walls are represented in black. The first exercise deals with the graphical interface and the second with oriented object programming aspects. The exercises are not independent but can be completed in any order. However, it is highly recommended to read the whole text before starting to answer.

Exercise 1 Graphical interface and events (14.5 marks)

The graphical interface is modelled by 3 JPanels : pMain, pDir and pTerrain, and 4 JButtons : bHaut, bBas, bGauche, bDroit. pTerrain and pDir are contained in pMain, and the 4 JButtons are contained in pDir. The coordinates of these elements are given in Figure 1 (right hand side). Colours of the JPanels pMain, pDir and pTerrain are yellow, green and white respectively.

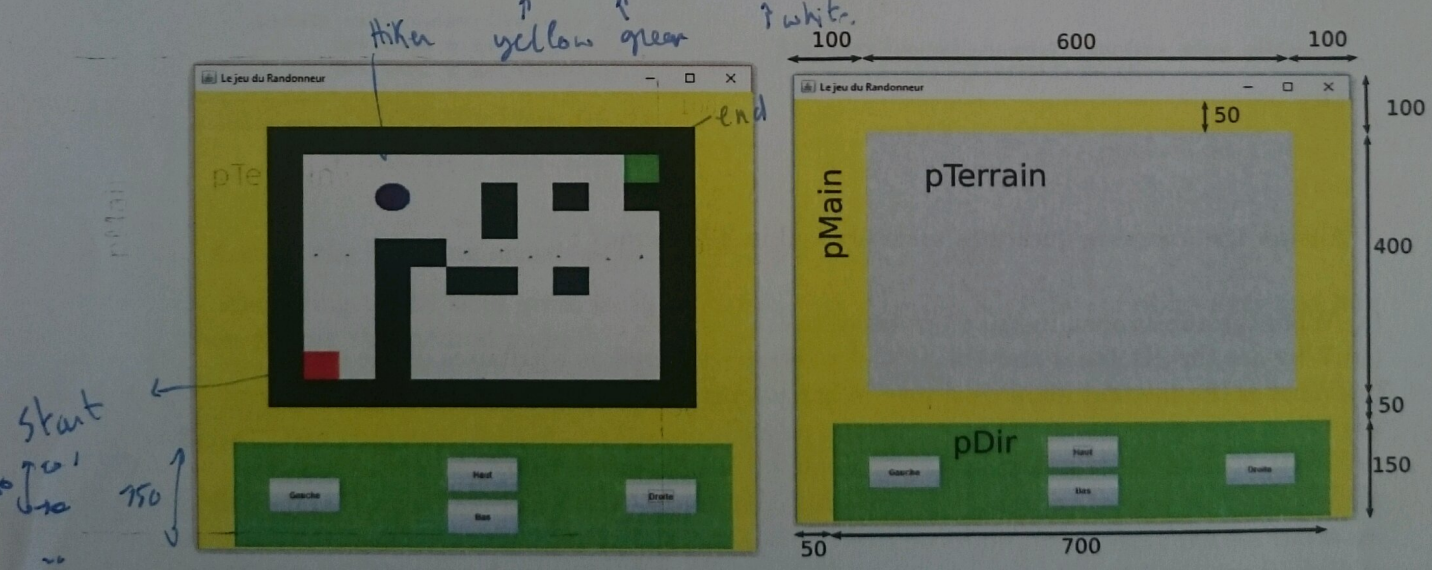


FIGURE 1 – Left : Graphical interface of the game. Right : Structure and position of the graphical components.

(Q1.1) Setting up of the graphical components – A few questions (3 marks)

The class Fenetre, partially detailed below, models the graphical interface.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Fenetre extends JFrame implements ActionListener{

    private JButton bHaut;
    private JButton bBas;
    private JButton bGauche;
    private JButton bDroite;

    private JPanel pTerrain;

    public Fenetre() {
          super("Le jeu du Randonneur");
          this.setSize(800,700);
          pTerrain = new Terrain();
          pTerrain.setLayout(null);
          JPanel pDir=new JPanel();
          pDir.setLayout(null);
          JPanel pMain = new JPanel();
          pMain.setLayout(null);
          bHaut   = new JButton("Haut");
          bBas    = new JButton("Bas");
          bGauche = new JButton("Gauche");
          bDroite = new JButton("Droite");
          bHaut  .setBounds(300,20,100,50);
          bBas   .setBounds(300,80,100,50);
          bGauche.setBounds(50,50,100,50);
          bDroite.setBounds(550,50,100,50);

          ...TO BE COMPLETED (Q1.2)...

    }


    public static void main(String[] a){
          new Fenetre();
    }

     public void actionPerformed(ActionEvent e){
                ...TO BE COMPLETED (Q1.8)...
    }
}
```

Answer the following questions precisely and in 3 lines max :

(a) What are the `import` instructions used for ?
(b) Why are the JButtons and the JPanel pTerrain declared as attributes of the class ?
(c) What is the method main used for ? What happens if it is removed ?
(d) What does `implement ActionListener` mean ?
(e) What does the instruction `super("Le jeu du Randonneur")` do ?
(f) Why is there a method `actionPerformed` ? What is it used for ?

(Q1.2) Setting up of the graphical components – Your turn to play (2.5 marks)

Complete the constructor of the class Fenetre in order to obtain the graphical interface displayed in figure 1. Sizes are given in pixels. The user needs to be able to interact with the buttons.

Note : to set the background colour of a JPanel p, e.g. to white, one needs to call p.setBackground(Color, WHITE);

(Q1.3) *Drawing in a JPanel. A few questions (1 mark)*

To simplify the interactions with the gameboard pTerrain, we are going to replace its type JPanel by the type Terrain defined below. The line private JPanel pTerrain in the class Fenetre is thus replaced by private Terrain pTerrain. The gameboard to be displayed is modelled by a matrix of integers between 0 and 3. The meaning of these integers is given by the constants CASE_VIDE, CASE_MUR, etc. The class Terrain has an attribute r of type Randonneur which represents the character to move. This class Randonneur will be studied more in detail in the next question.

```
public class Terrain extends JPanel {
    final public int CASE_VIDE = 0;
    final public int CASE_MUR = 1;
    final public int CASE_DEBUT = 2;
    final public int CASE_FIN = 3;

    // Creating the gameboard
    final private int[][] CARTE = {
        {1,1,1,1,1,1,1,1,1,1,1},
        {1,0,0,0,0,0,0,0,0,3,1},
        {1,0,0,0,0,0,1,1,0,1,1},
        {1,0,0,0,0,0,1,0,0,0,1},
        {1,0,0,1,1,0,0,0,0,0,1},
        {1,0,0,1,0,1,1,1,0,0,1},
        {1,0,0,1,0,0,0,0,0,0,1},
        {1,2,0,1,0,0,0,0,0,0,1},
        {1,1,1,1,1,1,1,1,1,1,1}
    };

    public Randonneur r;

    public Terrain() {
        r=new Randonneur(CARTE);
    }

    public void paint(Graphics g) {
        int haut = getHeight();
        int larg = getWidth();

        ...TO BE COMPLETED (Q1.4)...

        r.dessiner(g,larg,haut);
    }
}
```

Answer the following questions precisely and in 3 lines max :

(a) According to the sizes given in figure 1, what are the values of the variables haut and larg when exiting the method paint ?
(b) What is the method paint used for and when is it called ?

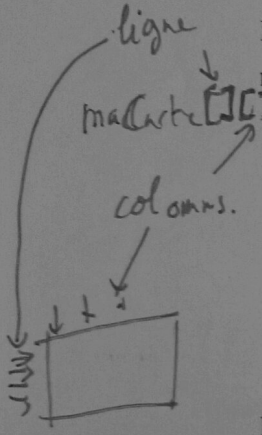
(Q1.4) *Drawing in a JPanel – Your turn to play (2,5 marks)*

Complete the method paint in the class Terrain. This method must display a rectangle for each cell of the gameboard. The colour of a cell depends on what it contains : red for the start cell (Color.RED), green for the finish cell (Color.GREEN), black for the walls (color.BLACK), and white for the empty cells (Color.WHITE).

(Q1.5) *The class Randonneur – A few questions (1,5 marks)*

Part of the class Randonneur is detailed below. That class represents the character which is going to be moved by the player by using buttons, and it updates the display. Its attributes x and y represent the character's column and row numbers in the gameboard (starting from the top) respectively. For the example provided in figure 1, x = 3 and y= 2.

```
public class Randonneur {
  private int x;
  private int y;
  private int[][] maCarte;
  private int ptVie;

  public Randonneur(int[][] carte) {
    //searching on the board for coordinates of a starting point in order to initialise x and y
    ...TO BE COMPLETED (Q1.6)...

    // initialisation of the other attributes
    maCarte=carte;
    ptVie=10;
  }

  public void dessiner(Graphics g, int w, int h) {
    int nbLignes = maCarte.length;
    int nbCols = maCarte[0].length;

    ...TO BE COMPLETED (Q1.7)...
  }

  public void marcher(int deltaX, int deltaY) {
    int nbLignes = maCarte.length;
    int nbCols = maCarte[0].length;
    int newx=x+deltaX;
    int newy=y+deltaY;

    if(maCarte[newy][newx] == 1) {
      System.out.println("Obstacle. On ne bouge pas.");
    } else {
        x=newx;
        y=newy;
        if(maCarte[y][x] == 3) {
          System.out.println("Bravo, le randonneur est sorti!");
        }
      }
  }

  public int getX() {return x;}
  public int getY() {return y;}
    ...
}
```



Answer the following questions precisely and in 3 lines max :

(a) What are the method getX and getY useful for ?
(b) What does the method marcher do ?
(c) With the provided data, what are the values of nbLignes and nbCols of the method marcher ?

(Q1.6) The class Randonneur – Your turn to play 1/2 (1 mark)

Complete the constructor of the class Randonneur in order to initialise the x and y coordinates by looking for the start cell (i.e. the cell equal to 2) in the board. Be aware that this method must work for any board and not only for the one provided as example.

(Q1.7) The class Randonneur – Your turn to play 2/2 (1 mark)

Give the implementation of the method dessiner in the class Randonneur. Let recall that this method draws a blue ellipse representing the hicker and is called by the method paint of the class Terrain (refer to the code of the class Terrain above).

(Q1.8) The method actionPerformed (2 marks)

Give the implementation of the method actionPerformed in the class Fenetre. This method moves the hicker one cell away (to the top, the bottom, the right or the left) through the method marcher, by a click on one of the buttons.

Exercise 2 Object Oriented Programming and polymorphism (9.5 marks)

We now want to improve the game by adding special objects in the maze. These objects are of 3 kinds :

- The mushrooms (Mushroom) add 2 life points to the hicker. They are represented by pink rectangles.
- The wildboars (WildBoar) take 3 life points away from the hicker. They are represented by grey rectangles.
- The magic hole (MagicHole) teleports the hicker onto a free cell. It is represented by a yellow rectangle.

These objects are organised in a class hierarchy illustrated in figure 2. The way these different objects are displayed is also illustrated on the left hand side of this figure.

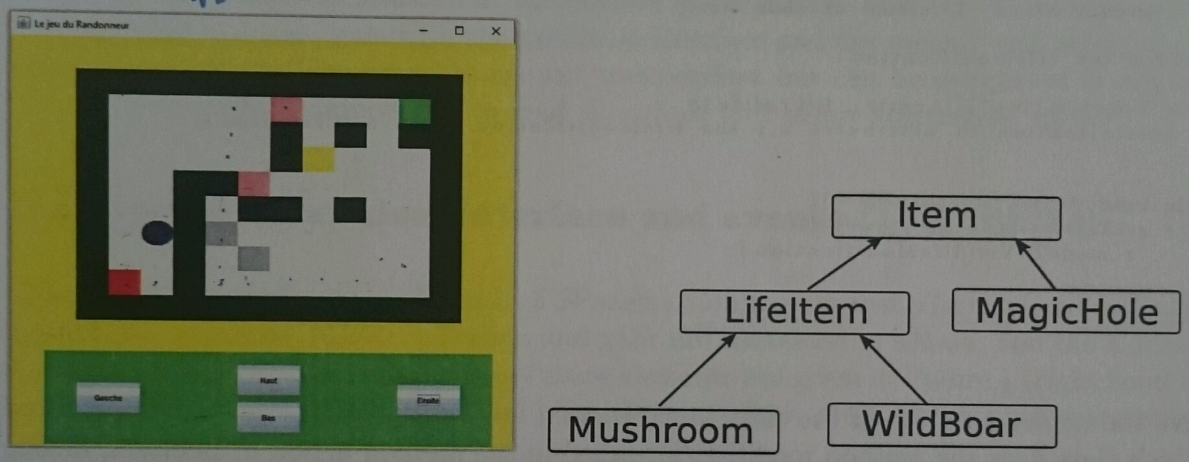


FIGURE 2 – Left : Graphical interface of the game with special objects. Right : Class hierarchy.

(Q2.1) Item and MagicHole (4 marks)

The class Item is partly detailed below.

```
public ... class Item {

  protected Color maCouleur;
  protected int x,y;
  private int nbLignes;
  private int nbCols;
  public Item(int[][] carte){
    nbLignes = carte.length;
    nbCols = carte[0].length;
    ...TO BE COMPLETED (Q2.1)...
  }

  public abstract void action(Randonneur r);

  public void dessiner(Graphics g,int w, int h){
        ...drawing a rectangle with maCouleur (code not required)...

  }
}
```

(a) Is the class Item an abstract class ? Justify your answer.
(b) Complete the constructor of the class Item. This constructor determines a position (x,y) by randomly choosing a cell on the gameboard provided as a parameter. It is important to notice

that this cell must be empty.

> **Note :** `(int)(x * Math.random())` returns a random integer belonging to [0, x-1]

(c) Give the implementation of the constructor of the class MagicHole which inherits from the class Item.
(d) Does the class MagicHole contain an implementation of the method action ? If so, provide its header (its implementation is not required).

(Q2.2) LifeItem and Mushroom (2 marks)

The class LifeItem is the ancestor of the classes that modify the hicker's life points (MushRoom and WildBoar). Part of its implementation is given below.

```
public abstract class  LifeItem extends Item{

  protected int lifeModification;

  public LifeItem(int[][] carte, int alife){
    ...Initialisation of attributes x,y and lifeModification (code not required)...
  }

  public void Action(Randonneur r){
    if(r.getX()==x && r.getY()==y){
          r.modifPtVie(lifeModification);
    }
  }
}
```

(a) Give the implementation of the class MushRoom.
(b) Which class does the method modifPtVie, called in the method action in LifeItem, belong to ? Give its header and its implementation.

(Q2.3) The array of Items (3,5 marks)

The special objects present in the game are modelled by an array of objects of the class Item declared as an attribute of the class Terrain : Item [] listItem. (a) What needs to be added in the method paint of the class Terrain in order to display those special objects ?
In question (a), we assume that the method dessiner of the class Item already exists.
(b) In which class and which method would you call the methods action associated to the special objects ? What code would need to be added ?