

Année universitaire 2016-2017
Matière : Programmation JavaNom : Prénom :
CIN : Groupe :

Nombres feuilles :

Signatures des surveillants :

Enseignant(es) : Iyed Ben Slimen & Sana Nouira,
Filière / Classe : 2^{ème} ING info
Nbre. de pages : 7 pages**Date** : 02/01/2017
Durée : 1h30
Calculatrice /document : non aut

Note :

Il sera tenu compte de la clarté et de la présentation des réponses.
Tout est clair. Aucune explication ne sera délivrée au cours de l'épreuve.
Il faut rendre toutes les feuilles même si elles sont blanches

On souhaite définir un programme en java permettant de positionner et de déplacer plusieurs Robots sur une interface. Pour cela vous choisirez les packages, les interfaces, les classes et les collections ... qui vous semblent les mieux adaptées en vue de répondre aux critères ci-dessous. **Vous ajouterez toutes les méthodes qui vous semblent nécessaires pour le bon fonctionnement de cette application.**

1- Un Robot est caractérisé par : - un **nom**, une couleur (de type Color)

- une position : entiers **x** et **y**, sachant que **x** augmente en allant vers l'Est et **y** augmente en allant vers le Nord
- une direction : **direction** qui prend une valeur « Nord » « Est » « Sud » « Ouest »

Un robot peut se déplacer d'un seul pas (méthode **deplacer()** incrémente ou décrémente la position **x** ou **y**), peut tourner à droite (de 90°) pour changer de direction (si sa direction était « Nord » elle devient « Est »...) (méthode **droite()**), et peut afficher son état en détail avec l'instruction `System.out.println()`. Le nom, la position et la direction lui sont donnés au moment de sa création. Le nom est obligatoire, la position par défaut est (0,0) et la direction par défaut est « Est ».

2 – Il existe une seconde catégorie de Robot, les robots Nouvelle génération RobotNG qui savent faire la même chose que les Robots précédents. Ils savent en plus se déplacer de plusieurs pas en même temps (méthode **deplacer()** qui prend en paramètre aussi le nombre de pas), tourner à gauche (de 90° avec la méthode **gauche()**) et peuvent afficher leur état en détail avec l'instruction `System.out.println()`. Cette classe permettra par la suite de gérer des flux d'objets.

3- Définir une interface graphique permettant de saisir les caractéristiques d'un Robot : le nom, choisir une couleur parmi une liste, choisir la direction aussi, la position initiale et un bouton « Valider » permettant d'enregistrer les informations saisies dans un fichier à flux de caractères (vous gérez les exceptions nécessaires). Vous dessinerez la maquette de votre interface graphique puis vous implémenterez la classe Interface1

4 – Afin d’améliorer votre application vous allez définir une classe **Jeu** permettant de stoker un ensemble de robotNG dans une collection. Cette classe est caractérisée par un ensemble de RobotNG, des méthodes permettant d’enregistrer et de lire les objets de type RobotNG dans un fichier à flux d’objet (vous gérer et définir toutes les exceptions et les méthodes nécessaires).

Matière : Programmation Java

Nom :

Prénom :

CIN :

Groupe :

Signatures des surveillants :

5- On souhaite réaliser une animation entre plusieurs RobotNG au maximum 5. Quelle modification doit t-on faire sur la classe RobotNG afin de lancer les robots en « parallèle », leur permettre de se déplacer de façon aléatoire (nombre de pas aléatoire compris entre 1 et 10) . Si le nombre de pas est inférieur à 5 la direction est gauche() sinon la direction est droite(). Redéfinir la classe RobotNG (ne pas réécrire les méthodes existantes mais juste les modifications nécessaires), définir un programme principal permettant de tester cette classe.

[illegible]

ANNEXE :

public class JButton public JButton (String); public class JTextField public JTextField (int); public void setText(String); String getText(); public class JLabel public JLabel (String); public class JComboBox public void addItem (String); int Integer.parseInt (String); public class GridLayout (nbr lignes, nbr colonnes, espacement lignes, espacement colonnes) ; public class Dimension (int,int) ; public class Graphics2D public void setpaint (Color) ; public setStroke(BasicStroke); public void draw (Shape); public void fill (Shape); public class Scanner () next(); nextInt();	public class JPanel public void add (Component); public void setVisible(boolean); public void paintCompent (Graphics); public void setBackground(Color); Graphics getGraphics(); public void setLayout(LayoutManager) public void setPreferredSize(Dimension); public class JFrame public JFrame (String); public void setTitle(String); public void add (Component); public void add (Component, BorderLayout.); public void setVisible(boolean); public void pack(); public BasicStroke(int); public class Ellipse2D //constreurs Ellipse2D.Float (int, int, int,int); Ellipse2D.Double(int,int,int,int);	public abstract interface ActionListener public void actionPerformed(ActionEvent e) public abstract interface ItemListener public void itemStateChanged(ItemEvent e) ItemEvent getStateChange() ItemEvent.SELECTED public abstract class MouseAdapter implements MouseListener public void mouseClicked (MouseEvent e) Méthodes de la classe Thread run() ; join() ; start() Boolean isAlive(); sleep (int) ; //temps courant : long System.currentTimeMillis()
--	---	--

Compléter sur une autre feuille si l'espace est insuffisant

Eléments de correction

1) Classe Robot :

```
import javafx.scene.paint.Color;
public class Robot {
    protected String nom;
    protected Color cl;
    protected int x,y;
    protected String direction;
    public Robot(String nom, Color cl, int x , int y, String direction){
        this.nom = nom;
        this.cl=cl;
        this.x=x; this.y=y;
        this.direction= direction;}

    public Robot(String nom, Color cl){
        this.nom = nom;
        this.cl=cl;
        x=0; y=0;
        direction= "EST";}

    public void deplacer(){
        switch (direction){
            case "EST":x++;
            case "OUEST": x--;
            case "NORD":y++;
            case "SUD":y--; }
        }

    public void droite () {
        switch (direction){
            case "EST": direction = "SUD";
            case "OUEST": direction = "NORD";
            case "NORD": direction = "EST" ;
            case "SUD": direction = "OUEST"; } }

    @Override
    public String toString(){
        return "le robot "+nom+" ayant la couleur "+cl+" est dirigé vers "+direction+" de coordonnées
        (" +x+", "+y+)"; }

    // getters-setters
    public void setNom (String nom){
        this.nom=nom; }
    public void setColor (Color cl){
        this.cl=cl; }
    public void setX (int x){
        this.x=x; }
    public void setY (int y){
        this.y=y; }
    public void setDirection(String direction){
        this.direction=direction; }
    public String getNom (){
        return nom; }
```



```

public Color getColor (){
    return cl; }
public int getX (){
    return x; }
public int getY (){
    return y; }
public String getDirection(){
    return direction; }}

```

2) Classe Robot Nouvelle Génération :

```

import javafx.scene.paint.Color;
import java.io.*;

```

```

public class RobotNG extends Robot implements Serializable {
    public RobotNG(String nom, Color cl, int x , int y, String direction){
        super(nom, cl, x , y, direction);}

```

```

    public RobotNG(String nom, Color cl){
        super(nom,cl);}
    public void deplacer(int pas){
        switch (direction){
            case "EST": x+=pas;
            case "OUEST": x-=pas;
            case "NORD":y+=pas;
            case "SUD":y-=pas; }
        }
    public void gauche () {
        switch (direction){
            case "EST": direction = "NORD";
            case "OUEST": direction = "SUD";
            case "NORD": direction = "OUEST" ;
            case "SUD": direction = "EST";}
        }
    }

```

@Override

```

    public String toString(){
        return "le robot "+nom+" ayant la couleur "+cl+" est dirigé vers "+direction+" de coordonnées ("
+x+", "+y+") il est de type NG";    }}

```

3) Interface (sous forme de maquette)

```

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import javafx.application.Application;

```

```

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.ListView;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

```

```

public class RobotInterface extends Application {

```

```

    VBox root;
    HBox h1, h2, h3;
    Label lb1,lb2,lb3,lb4;
    TextField t1,t2,t3;
    Button btn;
    ListView <String> l1,l2;
    String nomfichier;
    File f;

```

```

    @Override

```

```

    public void start(Stage primaryStage) throws IOException{
        root = new VBox();
        h1 = new HBox();
        h2 = new HBox();
        h3 = new HBox();
        lb1 = new Label("Nom");
        t1= new TextField();
        t1.setPrefColumnCount(10);
        t2= new TextField();
        t2.setPrefColumnCount(2);
        t3= new TextField();
        t3.setPrefColumnCount(2);
        lb2 = new Label("Couleur");
        lb3 = new Label("Position initiale");
        lb4 = new Label("Direction");
        btn = new Button("Valider");
        ObservableList<String> names = FXCollections.observableArrayList("Rouge", "Vert", "Bleu");
        ListView<String> l1 = new ListView<String>(names);
        l1.setPrefSize(100,20);
        ObservableList<String> names2 = FXCollections.observableArrayList("EST", "OUEST",
"NOT", "SUD");
        ListView<String> l2 = new ListView<String>(names2);
        l2.setPrefSize(100,20);
        h1.getChildren().addAll(lb1,t1,lb2,l1);
        h2.getChildren().addAll(lb3,t2,t3,lb4,l2);
        h3.getChildren().add(btn);
    }
}

```

```

File f = new File("robots.txt");
PrintWriter sortie = new PrintWriter(f);

btn.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent event) {

        sortie.println("nom: "+t1.getText()+" couleur:
"+l1.getSelectionModel().getSelectedItem()+" direction:
"+l2.getSelectionModel().getSelectedItem()+" position initiale:
("+t2.getText()+","+t3.getText()+")");
        sortie.close();
    }
});

root.getChildren().addAll(h1,h2,h3);
Scene scene = new Scene(root, 350, 100);
primaryStage.setTitle("Robots");
primaryStage.setScene(scene);
primaryStage.show();
}

public static void main(String[] args) {
    launch(args);
}
}

// remarque : vous pouvez changer les conteneurs ; par exemple en GridPane
4)
import java.util.ArrayList;
import java.io.*;

public class Jeu {
    String nom;
    ArrayList <RobotNG> li;
    ObjectOutputStream sortie;
    ObjectInputStream entree;
    public Jeu(String nom){
        this.nom = nom;
        li = new ArrayList(); }

    public void stocker(RobotNG Rn){
        li.add(Rn); }

    public void enregistrer (RobotNG Rn)throws IOException {
        sortie = new ObjectOutputStream(new FileOutputStream("RobotNG.dat"));
        sortie.writeObject(Rn);
        sortie.close(); }
}

```

```

public void lire() throws ClassNotFoundException, IOException{
int i=0; boolean eof = false; RobotNG Rn;
entree = new ObjectInputStream(new FileInputStream("RobotNG.dat"));
while (!eof){
    try{ i++;
        Rn = (RobotNG) entree.readObject();
        System.out.println("le robot nouvelle génération numéro "+i+": "+ Rn);    }
    catch(EOFException e){eof=true;}    }
entree.close();    }}

```

5) // Parallélisme :

```

import javafx.scene.paint.Color;
import java.io.*;
import static java.lang.Thread.interrupted;

```

```

public class RobotNG extends Robot implements Serializable, Runnable {
    public RobotNG(String nom, Color cl, int x , int y, String direction){
        super(nom, cl, x , y, direction);}

```

```

    public RobotNG(String nom, Color cl){
        super(nom,cl);}
    public int [] deplacer(int pas, int x, int y, String direct){
        switch (direct){
            case "EST": x+=pas;
            case "OUEST": x-=pas;
            case "NORD":y+=pas;
            case "SUD":y-=pas; }
        int [] res = {x,y};
        return res;
    }

```

```

    public String gauche (String direction) {
        String direction1=direction;
        switch (direction){
            case "EST": direction1 = "NORD";
            case "OUEST": direction1 = "SUD";
            case "NORD": direction1 = "OUEST" ;
            case "SUD": direction1 = "EST";}
        return direction1; }

```

```

    public String droite (String direction) {
        String direction1=direction;
        switch (direction){
            case "EST": direction1 = "SUD";
            case "OUEST": direction1 = "NORD";
            case "NORD": direction1 = "EST" ;
            case "SUD": direction1 = "OUEST"; }
        return direction1;
    }

```

@Override

```

    public String toString(){

```

```

        return "le robot "+nom+" ayant la couleur "+cl+" est dirigé vers "+direction+" de coordonnées (" +x+", "+y+") il est de type NG";
    }

```

```

@Override
public void run() {
    int x0= getX();
    int y0=getY();
    String direct = getDirection();
    System.out.println(this);
    while(!interrupted()){// boucle infinie
        //if (interrupted())return;
        //else {
        int d = (int) Math.random();
        int p = d+10;
        int[] tab = deplacer(p,x0,y0,direct);
        x0=tab[0];
        y0=tab[1];
        if(p<5) direct= gauche(direct); else direct = droite(direct);}
        System.out.println("le robot: "+getNom()+" est actuellement de direction "+direct+" de coordonnées (" +x0+", "+y0+")");
    }
}

```

```

import java.util.Scanner;
import javafx.scene.paint.Color;

```

```

public class Test {

    public static void main(String[] args) {
        Thread t1 = new Thread (new RobotNG("R1",Color.BLUE,12,3,"EST"));
        t1.start();
        Thread t2 = new Thread (new RobotNG("R2",Color.RED,1,5,"NORD"));
        t2.start();
        Scanner x = new Scanner(System.in);
        x.nextInt();
        t1.interrupt();
        Scanner x1 = new Scanner(System.in);
        x1.nextInt();
        t2.interrupt();
    }
}

```