DUT Info2 **2016/2017**
Date de rendu : 30 / 10 / 2017
Groupe TD : **4**
Nom : **GICQUEL Antoine, Julia Redor**

TP4 : Tâches :

L’objectif du TP est de manipuler des tâches Java en animant des objets sur un panneau.

Question : Écrivez une classe panneau qui crée un espace pour afficher des rectangles (largeur, hauteur).
Donnez le code et les traces.

Réponse :

```java
import javax.swing.*;
import java.awt.*;
public class Panneau extends JPanel {
    private int width;
    private int height;
    private int x;
    private int y;
    public Panneau (int width, int height) {
        this.width = width;
        this.height = height;
        Dimension size = new Dimension (width, height);
        setMinimumSize(size);
        setPreferredSize(size);
        setDoubleBuffered(true);
        x = 10;
        y = 10;
    }
    public synchronized void update () {
        repaint ();
    }
    public void paint (Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        g2.setColor (Color.GREEN);
        g2.fillRect (x, y, width, height);
    }
}
```

Question : Écrivez une classe pour créer un rectangle (couleur, largeur, hauteur). Et affichez-en plusieurs sur le panneau.
Donnez le code et les traces.

Réponse :

```java
import java.awt.*;
public class Tableau {
    private int width;
    private int height;
    private Color color;
    private int x, y;
    public Tableau(int width, int height, Color color) {
        this.width = width;
        this.height = height;
        this.color = color;
    }
    public int getWidth() {
        return width;
    }
    public void setWidth(int width) {
        this.width = width;
    }
    public int getHeight() {
        return height;
    }
    public void setHeight(int height) {
        this.height = height;
    }
    public Color getColor() {
        return color;
    }
    public void setColor(Color color) {
        this.color = color;
    }
    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }
}

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
public class Panneau extends JPanel {
    private int width;
    private int height;
    ArrayList<Tableau> tableaux;
    public Panneau (int width, int height) {
        this.width = width;
        this.height = height;
        Dimension size = new Dimension (width, height);
        setMinimumSize(size);
        setPreferredSize(size);
        setDoubleBuffered(true);
        tableaux = new ArrayList<>();
        Tableau t1 = new Tableau(300, 200, Color.BLUE);
        t1.setX(30);
        t1.setY(40);
        Tableau t2 = new Tableau(30, 20, Color.RED);
        t2.setX(5);
        t2.setY(10);
        tableaux.add(t1);
        tableaux.add(t2);
        new Thread(new Runnable() {
            @Override
            public void run() {
                while(true) {
                    update();
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }).start();
    }
    public synchronized void update () {
        repaint ();
    }
    public void paint (Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        for(Tableau t : tableaux) {
            g2.setColor(t.getColor());
            g2.fillRect(t.getX(), t.getY(), t.getWidth(), t.getHeight());
        }
    }
}
```

Question : Créez une activité pour déplacer chaque rectangle (vitesse horizontale, vitesse verticale). Lorsque le rectangle touche le bord gauche (ou haut) sa vitesse horizontale (ou verticale) devient positif, et réciproquement sur le bord droit (ou bas) elle devient négative.
Donnez le code et les traces.

→ Vérification avec un processus pour chaque rectangle.

Réponse :

```
import javax.swing.*;
import java.awt.*;
public class Tableau implements Runnable {
    Panneau panneau;
    private int width;
    private int height;
    private Color color;
    private int x, y;
    private int speedH;
    private int speedV;
    public Tableau(Panneau panneau, int x, int y, int width, int height, Color
color) {
        this.panneau = panneau;
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.color = color;
        speedH = 0;
        speedV = 0;
        new Thread(this).start();
    }
    @Override
    public void run() {
        while(true) {
            if (x == 0 || x + width == panneau.getWidth()) {
                speedH *= -1;
            }
            if (y == 0 || y + height == panneau.getHeight()) {
                speedV *= -1;
            }
```

```java
            x += speedH;
            y += speedV;
            try {
                Thread.sleep(Panneau.MILLIS_WAIT);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public int getWidth() {
        return width;
    }
    public void setWidth(int width) {
        this.width = width;
    }
    public int getHeight() {
        return height;
    }
    public void setHeight(int height) {
        this.height = height;
    }
    public Color getColor() {
        return color;
    }
    public void setColor(Color color) {
        this.color = color;
    }
    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }
    public void setSpeedH(int speedH) {
        this.speedH = speedH;
    }
    public void setSpeedV(int speedV) {
        this.speedV = speedV;
    }
    public int getSpeedH() {
        return speedH;
    }
    public int getSpeedV() {
        return speedV;
    }
}

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
public class Panneau extends JPanel implements Runnable {
    public final static int MILLIS_WAIT = 100;
    private int width;
    private int height;
    ArrayList<Tableau> tableaux;
    public Panneau (int width, int height) {
        this.width = width;
        this.height = height;
        Dimension size = new Dimension (width, height);
        setMinimumSize(size);
        setPreferredSize(size);
        setDoubleBuffered(true);
        tableaux = new ArrayList<>();
        Tableau t1 = new Tableau(this, 30, 40, 50, 100, Color.BLUE);
        t1.setSpeedH(1);
        t1.setSpeedV(1);
        Tableau t2 = new Tableau(this, 5, 10, 30, 20, Color.RED);
        t2.setSpeedV(1);
        tableaux.add(t1);
        tableaux.add(t2);
        new Thread(this).start();
    }
    @Override
    public void run() {
        while(true) {
            update();
            try {
                Thread.sleep(MILLIS_WAIT);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public synchronized void update () {
        repaint();
    }
    public void paint (Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        g2.setBackground (Color.WHITE);
        g2.clearRect (0, 0, width, height);
        for(Tableau t : tableaux) {
            g2.setColor(t.getColor());
            g2.fillRect(t.getX(), t.getY(), t.getWidth(), t.getHeight());
        }
    }
}
```

Question : Ajoutez un identifiant à chaque rectangle et mémorisez les pixels occupés par chacun dans le panneau. Faites en sorte qu'un rectangle ne se déplace que si les pixels correspondants sont libres.
Donnez le code et les traces.

→ Il y a un processus qui vérifie les collisions entre chaque rectangle. Si ils se touchent, ils changent de direction.

Réponse :

```
import javax.swing.*;
import java.awt.*;
public class Tableau implements Runnable {
    Panneau panneau;
    private int width;
    private int height;
    private Color color;
    private int x, y;
    private int speedH;
    private int speedV;
    public Tableau(Panneau panneau, int x, int y, int width, int height, Color
color) {
        this.panneau = panneau;
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.color = color;
        speedH = 0;
        speedV = 0;
        new Thread(this).start();
    }
    public void flip() {
        speedH *= -1;
        speedV *= -1;
        x += speedH;
        y += speedV;
    }
    @Override
    public void run() {
        while(true) {
            if (x <= 0 || x + width >= panneau.getWidth()) {
                speedH *= -1;
            }
            if (y <= 0 || y + height >= panneau.getHeight()) {
                speedV *= -1;
            }
            x += speedH;
            y += speedV;
            try {
                Thread.sleep(Panneau.MILLIS_WAIT);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public boolean collision(Tableau t) {
        boolean ret = false;
        if (x < (t.getX()+t.getWidth()) && (x+width) > t.getX()
                && y < t.getY()+t.getHeight() && (y+height) > t.getY()) {
            ret = true;
        }
        return ret;
    }
    public int getWidth() {
        return width;
    }
    public void setWidth(int width) {
        this.width = width;
    }
    public int getHeight() {
        return height;
    }
    public void setHeight(int height) {
        this.height = height;
    }
    public Color getColor() {
        return color;
    }
    public void setColor(Color color) {
        this.color = color;
```

```java
    }
    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }
    public void setSpeedH(int speedH) {
        this.speedH = speedH;
    }
    public void setSpeedV(int speedV) {
        this.speedV = speedV;
    }
    public int getSpeedH() {
        return speedH;
    }
    public int getSpeedV() {
        return speedV;
    }
}


import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
public class Panneau extends JPanel implements Runnable {
    public final static int MILLIS_WAIT = 100;
    private int width;
    private int height;
    ArrayList<Tableau> tableaux;
    public Panneau (int width, int height) {
        this.width = width;
        this.height = height;
        Dimension size = new Dimension (width, height);
        setMinimumSize(size);
        setPreferredSize(size);
        setDoubleBuffered(true);
        tableaux = new ArrayList<>();
        Tableau t1 = new Tableau(this, 30, 80, 100, 200, Color.BLUE);
        t1.setSpeedH(2);
        t1.setSpeedV(2);
        Tableau t2 = new Tableau(this, 5, 30, 60, 50, Color.RED);
        t2.setSpeedV(2);
        Tableau t3 = new Tableau(this, 300, 300, 100, 200, Color.GREEN);
        t3.setSpeedH(-4);
        t3.setSpeedV(-6);
        tableaux.add(t1);
        tableaux.add(t2);
        tableaux.add(t3);
        new Thread(this).start();
        enableCollision();
    }
    public void enableCollision() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                while(true) {
                    for(int i = 0; i < tableaux.size(); i++) {
                        Tableau t = tableaux.get(i);
                        for(int j = 0; j < tableaux.size(); j++) {
                            if(i != j && t.collision(tableaux.get(j))) {
                                //t.flip();
                                tableaux.get(j).flip();
                            }
                        }
                    }
                    try {
                        Thread.sleep(MILLIS_WAIT);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }).start();
    }
    @Override
    public void run() {
        while(true) {
            update();
        }
    }
    public synchronized void update () {
        repaint();
    }
    public void paint (Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        g2.setBackground (Color.WHITE);
        g2.clearRect (0, 0, width, height);
        for(Tableau t : tableaux) {
            g2.setColor(t.getColor());
            g2.fillRect(t.getX(), t.getY(), t.getWidth(), t.getHeight());
        }
    }
}
```

Question : Faites de même pour la création. Le rectangle ne doit apparaître que quand il n’y aura plus de rectangle là où on souhaite le créer.
Donnez le code et les traces.

→ Un processus lors de la création de chaque rectangle va vérifier si il ne gène pas les autres rectangles. Les rectangles ont maintenant des états visibles ou non.

Réponse :

```
import javax.swing.*;
import javax.swing.plaf.metal.MetalBorders;
import java.awt.*;
public class Tableau implements Runnable {
    Panneau panneau;
    private int width;
    private int height;
    private Color color;
    private int x, y;
    private int speedH;
    private int speedV;
    private boolean visible;
```

```java
    public Tableau(Panneau panneau, int x, int y, int width, int height, Color
color) {
        this.panneau = panneau;
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.color = color;
        speedH = 0;
        speedV = 0;
        visible = false;
        Thread moove = new Thread(this);
        final Tableau myself = this;
        new Thread(new Runnable() {
            @Override
            public void run() {
                boolean ok = false;
                while(!ok) {
                    ok = true;
                    for(Tableau t : panneau.getTableaux()) {
                        if(t.isVisible() && myself.collision(t))
                            ok = false;
                    }
                    if(!ok) {
                        try {
                            Thread.sleep(Panneau.MILLIS_WAIT);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
                visible = true;
                moove.start();
            }
        }).start();
    }
    public void flip() {
        speedH *= -1;
        speedV *= -1;
        x += speedH;
        y += speedV;
    }
    @Override
    public void run() {
        while(true) {
            if (x <= 0 || x + width >= panneau.getWidth()) {
                speedH *= -1;
            }
            if (y <= 0 || y + height >= panneau.getHeight()) {
                speedV *= -1;
            }
            x += speedH;
            y += speedV;
            try {
                Thread.sleep(Panneau.MILLIS_WAIT);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public boolean collision(Tableau t) {
        boolean ret = false;
        if (x < (t.getX()+t.getWidth()) && (x+width) > t.getX()
                && y < t.getY()+t.getHeight() && (y+height) > t.getY()) {
            ret = true;
        }
        return ret;
    }
    public int getWidth() {
        return width;
    }
    public void setWidth(int width) {
        this.width = width;
    }
    public int getHeight() {
        return height;
    }
    public void setHeight(int height) {
        this.height = height;
    }
    public Color getColor() {
        return color;
    }
    public void setColor(Color color) {
        this.color = color;
    }
    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }
    public void setSpeedH(int speedH) {
        this.speedH = speedH;
    }
    public void setSpeedV(int speedV) {
        this.speedV = speedV;
    }
    public int getSpeedH() {
        return speedH;
    }
    public int getSpeedV() {
        return speedV;
    }
    public boolean isVisible() {
        return visible;
    }
}

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
public class Panneau extends JPanel implements Runnable {
    public final static int MILLIS_WAIT = 100;
    private int width;
    private int height;
    ArrayList<Tableau> tableaux;
    public Panneau (int width, int height) {
        this.width = width;
        this.height = height;
        Dimension size = new Dimension (width, height);
        setMinimumSize(size);
        setPreferredSize(size);
        setDoubleBuffered(true);
        tableaux = new ArrayList<>();
        Tableau t1 = new Tableau(this, 30, 250, 100, 200, Color.BLUE);
        t1.setSpeedH(2);
        t1.setSpeedV(2);
        tableaux.add(t1);
        Tableau t2 = new Tableau(this, 5, 30, 60, 50, Color.RED);
        t2.setSpeedV(2);
        tableaux.add(t2);
        Tableau t3 = new Tableau(this, 300, 300, 100, 200, Color.GREEN);
        t3.setSpeedH(-4);
        t3.setSpeedV(-6);
        tableaux.add(t3);
        Tableau t4 = new Tableau(this, 30, 80, 50, 100, Color.BLACK);
        t4.setSpeedH(2);
        t4.setSpeedV(2);
        tableaux.add(t4);
        new Thread(this).start();
        enableCollision();
    }
    public void enableCollision() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                while(true) {
                    for(int i = 0; i < tableaux.size(); i++) {
                        Tableau t = tableaux.get(i);
                        for(int j = 0; j < tableaux.size(); j++) {
                            if(i != j && t.collision(tableaux.get(j))) {
                                //t.flip();
                                tableaux.get(j).flip();
                            }
                        }
                    }
                    try {
                        Thread.sleep(MILLIS_WAIT);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }).start();
    }
    @Override
    public void run() {
        while(true) {
            update();
        }
    }
    public synchronized void update () {
        repaint();
    }
    public void paint (Graphics g) {
        Graphics2D g2 = (Graphics2D) g;
        g2.setBackground (Color.WHITE);
        g2.clearRect (0, 0, width, height);
        for(Tableau t : tableaux) {
            if(t.isVisible()) {
                g2.setColor(t.getColor());
                g2.fillRect(t.getX(), t.getY(), t.getWidth(), t.getHeight());
            }
        }
    }
    public ArrayList<Tableau> getTableaux() {
        return tableaux;
    }
}
```

Question : Permettrez l’arrêter et la relance d’une activité (clic, touche clavier). Il ne devra y avoir qu’au maximum une activité par rectangle.
Donnez le code

→ Faisable avec un listener sur les le clavier, et en arrêtant la méthode run de tous les rectangles.

Réponse :
-