Rapport Gestion du Carrefoure

# Trinôme:

* GHODBANE Oussama G01
* BOUMAKRANE Hamza G01
* TEBRI Abdelhakim G01

# Dirigé par :

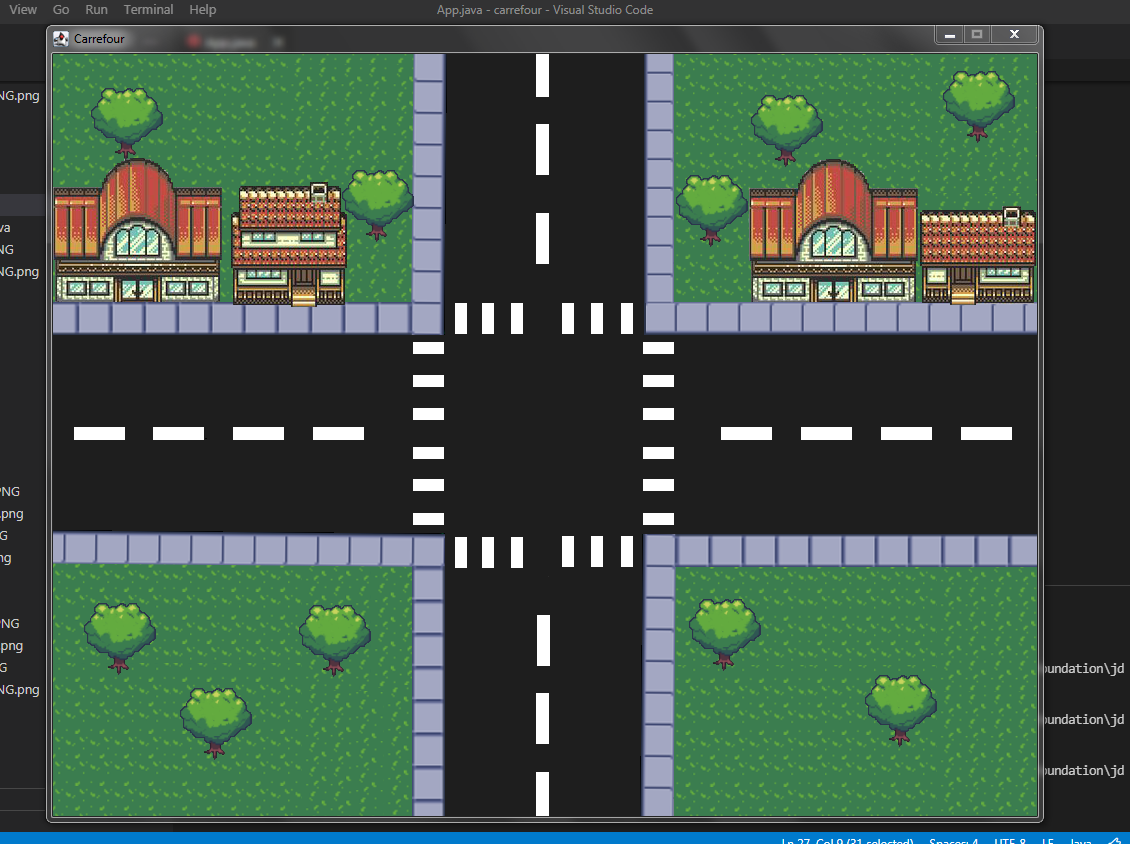
* Dr.BELGUIDOUM Meriem

Architecture de l'application

L’application contient 7 Classes (BackGround, XCar, YCar, XLight, YLight, Traffic, App) représentent les differentes composant du system .

# Class BackGround

Représente le plan du carrefour. Un BackGround est un JFrame ( BackGround extends JFrame ) responsable de la creation du fenetre de l’application, a un image d’un intersection come arriereplan.



# Class XLight

Représente les feux de circulation de la voie X, un feu est un “JLabel” (XLight extends JLabel),a deux icones pour un feu vert et un feu rouge, a un method toggleLight() qui bascule entre le feu vert et le feu rouge.

# Class YLight

Représente les feux de circulation de la voie Y, un feu est un “JLabel” (XLight extends JLabel),a deux icones pour un feu vert et un feu rouge, a un method toggleLight() qui bascule entre le feu vert et le feu rouge.



# Class XCar

Représente les véhicules qui circulent dans la voie X, une véhicule est un “JLabel RunnAble” ( XCar extends JLabel implements Runnable ), a des icones des differente véhicules, chaque instance choisis un icone aléatoire,

a 3 methods :

moveForward() : déplace la véhicule jusqu’à la point de feu de circulation.

crossIntersection() : traverse l’intersection.

keepMoving(): déplace la véhicule jusqu’à la fin du voie X.

les 3 methods sont appelées dans la method run().

# Class YCar

Représente les véhicules qui circulent dans la voie Y, une véhicule est un “JLabel RunnAble” ( YCar extends JLabel implements Runnable ), a des icones des differente véhicules, chaque instance choisis un icone aléatoire,

a 3 methods :

moveForward() : déplace la véhicule jusqu’à la point de feu de circulation.

crossIntersection() : traverse l’intersection.

keepMoving(): déplace la véhicule jusqu’à la fin du voie Y.

les 3 methods sont appelées dans la method run().



# Class Traffic

C’est un Thread ( Traffic extends Thread ) qui allume les feux de circulation, a la method run() qui appelle les methods toggleLight() de chaque feu de circulation.

# Class App

Contenant la method main(), elle est responsable de la declaration, l’instanciation et l’organisation des differente composent, a un boucle infini pour la creation des véhicule.

Les Semaphores utilisés

# Semaphore light\_x

Assure l’exclusion mutuelle de l’accès à l’intersection à partir de la voie X, initialisé a “1”.

# Semaphore light\_Y

Assure l’exclusion mutuelle de l’accès à l’intersection à partir de la voie Y, initialisé a “0”.

# Semaphore[ ] mutex\_x

Un tableau de 5 semaphores initialisé a “1”, pour simuler une file d’attente des vehicules avec accès FIFO dans la voie X.

# Semaphore[ ] mutex\_Y

Un tableau de 5 semaphores initialisé a “1”, pour simuler une file d’attente des vehicules avec accès FIFO dans la voie Y.

Le Code

# BackGround

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.UIManager;

/\*\*

 \* BackGoround

 \*/

public class BackGround extends JFrame {

    public BackGround(String name) {

        super(name);

        try {

            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

        }

        catch (Exception e) {

            System.err.println(e.getMessage());

        }

        this.setSize(1000, 800);

        this.setLocationRelativeTo(null);

        this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        this.setResizable(false);

        this.setContentPane(new JLabel(new ImageIcon(this.getClass().getResource("BackGround.PNG.png"))));

        this.getContentPane().setLayout(null);

        this.setVisible(true);

    }

}

## XLight

import javax.swing.ImageIcon;

import javax.swing.JLabel;

public class XLight extends JLabel {

    final static ImageIcon x\_green\_light = new ImageIcon(XLight.class.getClassLoader().getResource("X\_GreenLight2.PNG"));

    final static ImageIcon x\_red\_light = new ImageIcon(XLight.class.getClassLoader().getResource("X\_RedLight2.PNG"));

    final static int X\_LIGHT\_X = 606;

    final static int X\_LIGHT\_Y = 235;

    boolean currentLight = true;

    public XLight() {

        super(x\_green\_light);

        this.setSize(90, 30);

        this.setLocation(X\_LIGHT\_X, X\_LIGHT\_Y);

    }

    public void toggleLight() {

        if (this.currentLight) {

            this.setIcon(x\_red\_light);

            this.currentLight = false;

        } else {

            this.setIcon(x\_green\_light);

            this.currentLight = true;

        }

    }

}

## Ylight

import javax.swing.ImageIcon;

import javax.swing.JLabel;

public class YLight extends JLabel {

    final static ImageIcon y\_green\_light = new ImageIcon(XLight.class.getClassLoader().getResource("Y\_GreenLight2.PNG"));

    final static ImageIcon y\_red\_light = new ImageIcon(XLight.class.getClassLoader().getResource("Y\_RedLight2.PNG.png"));

    final static int Y\_LIGHT\_X = 346;

    final static int Y\_LIGHT\_Y = 175;

    boolean currentLight = false;

    public YLight() {

        super(y\_red\_light);

        this.setSize(30, 90);

        this.setLocation(Y\_LIGHT\_X, Y\_LIGHT\_Y);

    }

    public void toggleLight() {

        if (!this.currentLight) {

            this.setIcon(y\_green\_light);

            this.currentLight = true;

        } else {

            this.setIcon(y\_red\_light);

            this.currentLight = false;

        }

    }

}

## XCar

import java.util.Random;

import java.util.concurrent.Semaphore;

import javax.swing.ImageIcon;

import javax.swing.JLabel;

public class XCar extends JLabel implements Runnable {

    Semaphore[] mutex\_x;

    Semaphore light\_x;

    Random random = new Random();

    ImageIcon Xcars[] = { new ImageIcon(XCar.class.getClassLoader().getResource("TruckX.PNG")) ,

            new ImageIcon(XCar.class.getClassLoader().getResource("BusX.PNG")),

            new ImageIcon(XCar.class.getClassLoader().getResource("CarX.PNG")),

            new ImageIcon(XCar.class.getClassLoader().getResource("PoliceX.PNG"))};

    final static int LIGHT\_ACQUIRE = 606;

    final static int LIGHT\_RELEASE = 320;

    final static int Y\_AXIS = 300;

    //int x\_axis = 918;

    int x\_axis = 985 ;

    public XCar(Semaphore light\_x, Semaphore[] mutex\_x) {

        super();

        this.light\_x = light\_x;

        this.mutex\_x = mutex\_x;

        this.setIcon(Xcars[random.nextInt((3)+1)]);

        this.setSize(93, 46);

        this.setLocation(x\_axis, Y\_AXIS);

    }

      public void moveForward() throws InterruptedException{

        for (int i=4 ; i>=0;i--){

            mutex\_x[i].acquire();

            if (i<4) mutex\_x[i+1].release();

            while (x\_axis > LIGHT\_ACQUIRE+95\*i) {

                x\_axis--;

                this.setLocation(x\_axis, Y\_AXIS);

                try {

                    Thread.sleep(7);

                } catch (InterruptedException e) {}

            }

        }

     }

    public void crossIntersection() {

        mutex\_x[0].release();

        while (x\_axis > LIGHT\_RELEASE) {

            x\_axis--;

            this.setLocation(x\_axis, Y\_AXIS);

            try {

                Thread.sleep(7);

            } catch (InterruptedException e) {

                //TODO: handle exception

            }

        }

    }

    public void keepMoving() {

        while (x\_axis > -100) {

            x\_axis--;

            this.setLocation(x\_axis, Y\_AXIS);

            try {

                Thread.sleep(7);

            } catch (InterruptedException e) {}

        }

    }

    public void run() {

        try {

            moveForward();

            light\_x.acquire();

            crossIntersection();

            light\_x.release();

            keepMoving();

        } catch (InterruptedException e1) {

            // TODO Auto-generated catch block

            e1.printStackTrace();

        }

    }

}

## YCar

import java.util.Random;

import java.util.concurrent.Semaphore;

import javax.swing.ImageIcon;

import javax.swing.JLabel;

public class YCar extends JLabel implements Runnable {

    Semaphore[] mutex\_y;

    Semaphore light\_y;

    Random random = new Random();

    ImageIcon Ycars[] = { new ImageIcon(XCar.class.getClassLoader().getResource("CarY1.PNG")) ,

            new ImageIcon(XCar.class.getClassLoader().getResource("BusY.PNG")),

            new ImageIcon(XCar.class.getClassLoader().getResource("PoliceY.PNG")),

            new ImageIcon(XCar.class.getClassLoader().getResource("TruckY.PNG"))};

    final static int LIGHT\_ACQUIRE = 195;

    final static int LIGHT\_RELEASE = 481;

    final static int X\_AXIS = 411;

    //int y\_axis = -39;

    int y\_axis = -95;

    public YCar(Semaphore light\_y, Semaphore mutex\_y[]) {

        super();

        this.light\_y = light\_y;

        this.mutex\_y = mutex\_y;

        this.setIcon(Ycars[random.nextInt((3)+1)]);

        this.setSize(46,93);

        this.setLocation(X\_AXIS, y\_axis);

    }

    public void moveForward() throws InterruptedException{

        for (int i=3 ; i>=0;i--){

            mutex\_y[i].acquire();

            if (i<3) mutex\_y[i+1].release();

            while (y\_axis < LIGHT\_ACQUIRE-78\*i) {

              y\_axis++;

              this.setLocation(X\_AXIS, y\_axis);

              try {

                  Thread.sleep(7);

              } catch (InterruptedException e) {}

          }

        }

   }

    public void crossIntersection() {

        mutex\_y[0].release();

        while (y\_axis < LIGHT\_RELEASE) {

            y\_axis++;

            this.setLocation(X\_AXIS, y\_axis);

            try {

                Thread.sleep(7);

            } catch (InterruptedException e) {

                //TODO: handle exception

            }

        }

    }

    public void keepMoving() {

        while (y\_axis < 762) {

            y\_axis++;

            this.setLocation(X\_AXIS, y\_axis);

            try {

                Thread.sleep(7);

            } catch (InterruptedException e) {

                //TODO: handle exception

            }

        }

    }

    public void run() {

        try {

            moveForward();

            light\_y.acquire();

            crossIntersection();

            light\_y.release();

            keepMoving();

        } catch (InterruptedException e) {}

    }

}

# Traffic

import java.util.concurrent.Semaphore;

public class Traffic extends Thread{

    Semaphore light\_x;

    Semaphore light\_y;

    XLight xlight;

    YLight ylight;

    char greenLight = 'x';

    public Traffic(XLight xlight, YLight ylight, Semaphore light\_x, Semaphore light\_y) {

        this.xlight = xlight;

        this.ylight = ylight;

        this.light\_x = light\_x;

        this.light\_y = light\_y;

    }

    public void run() {

        while (true) {

            try {

                sleep(7000);

            } catch (InterruptedException e) {

                // TODO Auto-generated catch block

            }

            if (greenLight == 'x') {

                try {

                    light\_x.acquire();

                    light\_y.release();

                    xlight.toggleLight();

                    ylight.toggleLight();

                    greenLight = 'y';

                } catch (InterruptedException e) {

                    // TODO Auto-generated catch block

                    e.printStackTrace();

                }

            } else {

                try {

                    light\_y.acquire();

                    light\_x.release();

                    ylight.toggleLight();

                    xlight.toggleLight();

                    greenLight = 'x';

                } catch (InterruptedException e) {

                    e.printStackTrace();

                }

            }

        }

    }

}

# App

import java.util.concurrent.Semaphore;

public class App {

    public static void main(String[] args) throws Exception {

        System.out.println("Hello, World!");

        Semaphore[] mutex\_x = new Semaphore[5];

        for (int i = 0; i < mutex\_x.length; i++) {

            mutex\_x[i] = new Semaphore(1);

        }

        Semaphore[] mutex\_y = new Semaphore[4];

        for (int i = 0; i < mutex\_y.length; i++) {

            mutex\_y[i] = new Semaphore(1);

        }

        Semaphore light\_x = new Semaphore(1);

        Semaphore light\_y = new Semaphore(0);

        XLight xlight = new XLight();

        YLight ylight = new YLight();

        XCar xcar;

        YCar ycar;

        BackGround crossRoad = new BackGround("Carrefour");

        crossRoad.add(xlight);

        crossRoad.add(ylight);

        Traffic traffic = new Traffic(xlight, ylight, light\_x, light\_y);

        traffic.start();

        boolean flag = true;

        while (flag) {

            xcar = new XCar(light\_x, mutex\_x);

            crossRoad.add(xcar);

            new Thread(xcar).start();

            try {

                Thread.sleep((long)(Math.random()\*4000)+1);

            } catch (InterruptedException e) {}

            ycar = new YCar(light\_y, mutex\_y);

            crossRoad.add(ycar);

            new Thread(ycar).start();

            try {

                Thread.sleep((long)(Math.random()\*4000)+1);

            } catch (InterruptedException e) { }

        }

    }

}

ScreenShot



GitHub Link

<https://github.com/Ghoussama2001/crossroad_management>