1. -----------------------------------------Choptick-----------------------------------------

package diningPhilosophers;

public class Chopstick {

private int id; // Định số thứ tự của nĩa

private boolean taken = false; // Trạng thái của nĩa (đã được dùng hay chưa)

private int takenBy = -1; // Xác định số thứ tự của triết gia đang dùng nĩa (nếu có)

public Chopstick(int id) {

this.id = id;

}

synchronized void put() {

taken = false;

takenBy = -1;

notify();

}

synchronized void get() throws InterruptedException {

while(taken) wait();

taken = true;

}

public boolean isTaken() {

return taken;

}

public int getId() {

return id;

}

public int getTakenBy() {

return takenBy;

}

public void setTakenBy(int takenBy) {

this.takenBy = takenBy;

}

}

1. -----------------------------------------Diner-----------------------------------------

Ss

package diningPhilosophers;

public class Dinner {

private final static int PHILSNUM = 5;

private Philosopher[] philosophers = new Philosopher[PHILSNUM];

private Chopstick[] chopsticks = new Chopstick[PHILSNUM];

public Dinner() {

for(int i = 0; i < PHILSNUM; i++) {

// Khởi tạo 5 chiếc nĩa (tài nguyên găng)

chopsticks[i] = new Chopstick(i);

}

for(int i = 0; i < PHILSNUM; i++) {

// Khởi tạo các triết gia (các tiến trình)

philosophers[i] = new Philosopher(i, chopsticks[(i - 1 + PHILSNUM) % PHILSNUM], chopsticks[i]);

// Chạy tiến trình

philosophers[i].start();

}

}

public Philosopher[] getPhilosophers() {

return philosophers;

}

public Chopstick[] getChopsticks() {

return chopsticks;

}

public void stop() {

for(int i = 0; i< PHILSNUM; i++) {

philosophers[i].interrupt();

}

}

}

1. -----------------------------------------DinerGUI. -----------------------------------------

package diningPhilosophers;

import java.awt.Button;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Image;

import java.awt.Point;

import java.awt.Rectangle;

import java.awt.RenderingHints;

import java.awt.Scrollbar;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.geom.AffineTransform;

import java.awt.image.AffineTransformOp;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.net.URL;

import javax.imageio.ImageIO;

import javax.swing.JButton;

import javax.swing.JComponent;

import javax.swing.JFrame;

import javax.swing.JScrollBar;

public class DinnerGUI extends JFrame implements ActionListener {

private static final int n = 5;

Dinner dinner;

private Philosopher[] philosophers;

private Chopstick[] chopsticks;

int w = 1000, h = 700;

Point center = new Point(w/2 - 50,h/2 - 50);

int philoR = 220;

int chopR = 110;

int dishR = 110;

int tableR = 145;

Point[] philoPoints = new Point[n];

Point[] chopPoints = new Point[n];

Point[] dishPoints = new Point[n];

BufferedImage[] imgs = new BufferedImage[6];

double alpha = 2\*Math.PI/n;

double beta = alpha/2;

JButton btn;

public void setPoint() {

for(int i=0;i<5;i++) {

Point philoPoint = new Point((int)(center.x+philoR\*Math.cos(-i\*alpha)),(int)(center.y+philoR\*Math.sin(-i\*alpha)));

philoPoints[i] = philoPoint;

Point chopPoint = new Point((int)(center.x+chopR\*Math.cos(-(i\*alpha + beta))),(int)(center.y+chopR\*Math.sin(-(i\*alpha + beta))));

chopPoints[i] = chopPoint;

Point dishPoint = new Point((int)(center.x+dishR\*Math.cos(-i\*alpha)),(int)(center.y+dishR\*Math.sin(-i\*alpha)));

dishPoints[i] = dishPoint;

}

}

public void setImages() {

try {

imgs[0] = getBufferedImage("D:\\Java\\DoAnHeDieuHanh\\src\\diningPhilosophers\\Images\\eat.png");

imgs[1] = getBufferedImage("D:\\Java\\DoAnHeDieuHanh\\src\\diningPhilosophers\\Images\\think.png");

imgs[2] = getBufferedImage("D:\\Java\\DoAnHeDieuHanh\\src\\diningPhilosophers\\Images\\hungry.png");

imgs[3] = getBufferedImage("D:\\Java\\DoAnHeDieuHanh\\src\\diningPhilosophers\\Images\\getRight.png");

imgs[4] = getBufferedImage("D:\\Java\\DoAnHeDieuHanh\\src\\diningPhilosophers\\Images\\getLeft.png");

imgs[5] = getBufferedImage("D:\\Java\\DoAnHeDieuHanh\\src\\diningPhilosophers\\Images\\eatingIcon.png");

} catch (IOException e) {

e.printStackTrace();

}

}

public void InitPhilosophersAndChopsticks() {

dinner = new Dinner();

philosophers = dinner.getPhilosophers();

for(int i=0;i<n;i++)

philosophers[i].setView(this);

chopsticks = dinner.getChopsticks();

}

public DinnerGUI() {

this.setTitle("Dinner Philosophers");

this.setSize(w, h);

this.setLayout(null);

this.setLocation(450, 200);

this.setDefaultCloseOperation(3);

btn = new JButton("Pause");

btn.setBounds(20, 390, 100, 30);

btn.addActionListener(this);

this.add(btn);

setPoint();

setImages();

InitPhilosophersAndChopsticks();

this.setResizable(false);

this.setVisible(true);

}

@Override

public void paint(Graphics g) {

super.paint(g);

RenderingHints rh = new RenderingHints(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

rh.put(RenderingHints.KEY\_RENDERING, RenderingHints.VALUE\_RENDER\_QUALITY);

((Graphics2D) g).setRenderingHints(rh);

g.setColor(Color.BLACK);

Font myFont = new Font("Arial", Font.ITALIC, 20);

g.setFont(myFont);

g.drawImage(imgs[5], 20, 460, 64, 64, this);

g.drawString("EATING", 90, 500);

g.drawImage(imgs[1], 20, 530, 64, 64, this);

g.drawString("THINKING", 90, 570);

g.drawImage(imgs[2], 20, 600, 64, 64, this);

g.drawString("HUNGRY", 90, 640);

//draw table and dishes

g.setColor(Color.DARK\_GRAY);

g.fillOval(center.x - tableR + 60, center.y - tableR + 60, 2 \* tableR, 2\* tableR);

g.setColor(Color.WHITE);

for(int i = 0; i < n; i++) {

g.fillOval(dishPoints[i].x + 31, dishPoints[i].y + 31, 60, 60);

}

BufferedImage img;

for(int i=0;i<n;i++) {

if(philosophers[i].status == Status.THINKING) {

img = imgs[1];

}else if(philosophers[i].status == Status.HUNGRY) {

img = imgs[2];

}else if(philosophers[i].status == Status.GETRIGHT) {

img = imgs[3];

}else if(philosophers[i].status == Status.GETLEFT) {

img = imgs[4];

}else {

img = imgs[0];

}

double rotationRequired = Math.PI/2 - i\*alpha;

double locationX = img.getWidth()/2;

double locationY = img.getHeight()/2;

AffineTransform tx = AffineTransform.getRotateInstance(rotationRequired, locationX, locationY);

AffineTransformOp op = new AffineTransformOp(tx, AffineTransformOp.TYPE\_BILINEAR);

// Drawing the rotated image at the required drawing locations

g.drawImage(op.filter(img, null), philoPoints[i].x, philoPoints[i].y, this);

g.setColor(Color.BLACK);

myFont = new Font("Arial", Font.BOLD, 20);

g.setFont(myFont);

g.drawString("P" + i, philoPoints[i].x + 54, philoPoints[i].y + 135);

if(!chopsticks[i].isTaken()) {

g.setColor(Color.RED);

g.fillRect(chopPoints[i].x + 55, chopPoints[i].y + 40, 7, 50);

}

}

}

// Cập nhật vùng giao diện của triết gia có id = philoID

public void updatePhilosophers(int philoID) {

repaint(philoPoints[philoID].x, philoPoints[philoID].y, imgs[0].getWidth(), imgs[0].getHeight());

//delay để quan sát giao diện được tốt hơn

try {

philosophers[philoID].sleep(500);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

// Cập nhật vùng giao diện của nĩa có id = chopstickID

public void updateChopstick(int chopstickID) {

repaint(chopPoints[chopstickID].x + 55, chopPoints[chopstickID].y + 40, 7, 50);

}

public static void main(String[] args) {

new DinnerGUI();

}

@Override

public void actionPerformed(ActionEvent e) {

try {

if(e.getActionCommand() == "Pause") {

for(int i = 0; i < n; i++) {

philosophers[i].suspend();

// philosophers[i].interrupt();

}

btn.setText("Resume");

}

if(e.getActionCommand() == "Resume") {

for(int i = 0; i < n; i++) {

philosophers[i].resume();

// if(Thread.interrupted()) {

// continue;

// }

}

btn.setText("Pause");

}

}catch(Exception ex) {

ex.printStackTrace();

}

}

private Image getImage(String path) {

Image image = null;

try {

URL imageURL = DinnerGUI.class.getResource(path);

image = Toolkit.getDefaultToolkit().getImage(imageURL);

} catch (Exception e) {

System.err.println("Could not load image from path: " + path);

}

return image;

}

private BufferedImage getBufferedImage(String path) throws IOException {

File img = new File(path);

BufferedImage image = ImageIO.read(img);

return image;

}

}

1. -----------------------------------------Philosopher-----------------------------------------

package diningPhilosophers;

import java.util.Random;

public class Philosopher extends Thread {

private int id;

private Chopstick left;

private Chopstick right;

public Status status = Status.THINKING;

private DinnerGUI view;

private Random rand;

public Philosopher(int id, Chopstick l, Chopstick r) {

this.id = id;

this.left = l;

this.right = r;

rand = new Random();

}

private int generateRandomSleepAndThinkingTime() {

return rand.nextInt(5001) + 5000;

}

public void Wait() throws InterruptedException {

this.status = Status.HUNGRY;

view.updatePhilosophers(this.id);

if(id % 2 == 0) {

right.get();

right.setTakenBy(this.id);

this.status = Status.GETRIGHT;

view.updateChopstick(right.getId());

view.updatePhilosophers(this.id);

left.get();

left.setTakenBy(this.id);

this.status = Status.EATING;

view.updateChopstick(left.getId());

view.updatePhilosophers(this.id);

}else {

left.get();

left.setTakenBy(this.id);

this.status = Status.GETLEFT;

view.updateChopstick(left.getId());

view.updatePhilosophers(this.id);

right.get();

right.setTakenBy(this.id);

this.status = Status.EATING;

view.updateChopstick(right.getId());

view.updatePhilosophers(this.id);

}

}

public void Signal() {

right.put();

this.status = Status.THINKING;

view.updateChopstick(right.getId());

view.updatePhilosophers(this.id);

left.put();

view.updateChopstick(left.getId());

}

@Override

public void run() {

try {

while(true) {

//Thời gian triết gia suy nghĩ

//(Tiến trình ở ngoài vùng găng)

sleep(generateRandomSleepAndThinkingTime());

// Hành động lấy nĩa của triết gia khi đói

//(Tiến trình yêu cầu vào vùng găng)

Wait();

//Thời gian triết gia ăn

//(Tiến trình thực thi bên trong vùng găng)

sleep(generateRandomSleepAndThinkingTime());

//Hành động đặt nĩa xuống của triết gia khi ăn xong

//(Tiến trình ra khỏi vùng găng)

Signal();

}

}catch(InterruptedException e) {

e.printStackTrace();

}

}

public void setView(DinnerGUI view) {

this.view = view;

}

public Chopstick getLeft() {

return left;

}

public Chopstick getRight() {

return right;

}

}

1. -----------------------------------------Status-----------------------------------------

package diningPhilosophers;

public enum Status {

EATING,

THINKING,

HUNGRY,

GETRIGHT,

GETLEFT

}