# MÔN : CÔNG NGHỆ JAVA

## Bài thực hành 7.4 : Viết phần mềm giải quyết tương tranh giữa các threads

**I. Mục tiêu :**

- Giúp SV làm quen với việc dùng class Thread của package java.lang để quản lý thread.
- Giúp SV thấy được vấn đề tương tranh giữa các thread khi chúng cùng truy xuất tài nguyên dùng chung.
- Giúp SV làm quen với việc dùng class Semaphore của package java.lang để giải quyết tương tranh.
- Giúp SV thấy được hiện tượng dealock là hậu quả của việc dùng Semaphore giải quyết tương tranh giữa các thread.

**II. Nội dung :**

- Nâng cấp bài thực hành 7.3 để giải quyết tương tranh giữa các thread khi chúng cùng truy xuất vào cùng cell màn hình.
- Quan sát không còn hiện tượng icon của thread này đè mất icon của thread khác, bây giờ chúng phải chờ nhau. Tuy nhiên hiện tượng deadlock lại xảy ra, tần suất càng nhiều khi số lượng thread chạy càng lớn.

**III. Chuẩn đầu ra :**

- Sinh viên nắm vững và sử dụng thành thạo class Thread để quản lý thread.
- Sinh viên nắm vững và sử dụng thành thạo class Semaphore để giải quyết tương tranh giữa các thread khi chúng truy xuất tài nguyên dùng chung.

**IV. Qui trình :**

0. Copy thư mục quản lý Project được tạo ra bởi bài thực hành 7.3 thành thư mục NBThreadDemo3
1. Chạy NetBean 7.3.1, nếu cửa sổ Project có hiển thị các Project cũ hãy đóng chúng lại.
2. Chọn menu File.Open Project để máy hiển thị cửa sổ "Open Project", duyệt tìm thư mục NBThreadDemo3 để mở lại Project tương ứng.
3. Làm hiển thị cửa sổ soạn mã nguồn của class MyThread, dời cursor đến đầu các vị trí được tô màu dưới đây rồi thêm code được tô màu vào như sau :

```
package nbthreaddemo2;
//import các package cần dùng
import javax.swing.*;
import java.awt.*;
import java.util.*;
import java.awt.image.*;
import java.util.concurrent.TimeUnit;

//đặc tả class quản lý thread được dùng trong chương trình
public class MyThread extends Thread {
      //định nghĩa các hằng, thuộc tính cần dùng
      final double PI = 3.1416;
      final int xCell = 30;        //độ rộng của mỗi cell (pixel)
      final int yCell = 30;        //độ cao của mỗi cell (pixel)
      public Boolean fstart;       //trạng thái Start của thread
      public Boolean fstop;        //trạng thái Start của thread
```

```
  public Boolean fsuspend;          //trạng thái Suspend của thread
  public Boolean WaitOne = false;          //trạng thái chờ truy xuất cell
  public BufferedImage Pic;         //icon miêu tả thread
  int xCount;          //độ rộng vùng chạy của thread (số cell)
  int yCount;          //độ cao vùng chạy của thread (số cell)
  int tgchay;          //thời gian tính trước mỗi bước chạy của thread
  public Point Pos = new Point();  //vị trí của thread trong vùng chạy
  double dblGocChay;       //góc chạy của thread
  double tx, ty;       //bước tăng theo x và y
  MainForm frm;    //đối tượng quản lý Form ứng dụng
  //hàm khởi tạo các thông số của thread
  public MyThread(Random rnd, int xMax, int yMax, MainForm frm)
  {
     xCount = xMax; yCount = yMax; this.frm = frm;
     Pos.x = (int)(rnd.nextInt(xCount));
     Pos.y = (int)(rnd.nextInt(yCount));
     dblGocChay = ChinhGocChay(rnd.nextInt(360));
     tgchay = 1500+100*rnd.nextInt(100);
  }

  //định nghĩa hàm giả lập công việc tính toán của thread
  void MySleep(long count)
  {
     long i, j, k = 0;
     for (i = 0; i < count; i++)
        for (j = 0; j < 64000; j++) k = k + 1;
}

//tác vụ chức năng của thread
public void run() {
     int x1, y1;
     int x2, y2;
     int x, y;
     boolean kq = true;
     try  {
          //xin khóa truy xuất cell xuất phát (do thuộc tính Pos qui định)
          frm.mutList[Pos.y][Pos.x].acquire();
          while (fstart) { //lặp trong khi chưa có yêu cầu kết thúc
             //xác định tọa độ hiện hành của thread
             x1 = Pos.x; y1 = Pos.y;
             //hiển thị icon của thread ở (x1,y1)
             frm.gh.drawImage(Pic, xCell * x1, yCell * y1,null);
             //xác định màu vẽ hướng chạy của icon
             Color c = new Color(Pic.getRGB(1, 1),true);
             int yR, yG, yB;
             if (c.getRed() > 128) yR = 0; else yR = 255;
             if (c.getGreen() > 128) yG = 0; else yG = 255;
             if (c.getBlue() > 128) yB = 0; else yB = 255;
             frm.gh.setColor(new Color(yR, yG, yB));
             if (tx >= 0 && ty >= 0) { //hiện mũi tên góc dưới phải
```

```
                x = xCell * x1 + xCell - 2;
                y = yCell * y1 + yCell - 2;
                frm.gh.drawLine(x, y, x - 10, y);
                frm.gh.drawLine(x, y, x, y - 10);
            }
            else if (tx >= 0 && ty < 0) { //hiện mũi tên góc trên phải
                x = xCell * x1 + xCell - 2;
                y = yCell * y1 + 2;
                frm.gh.drawLine(x, y, x - 10, y);
                frm.gh.drawLine(x, y, x, y + 10);
            }
            else if (tx < 0 && ty >= 0) { //hiện mũi tên góc dưới trái
                x = xCell * x1 + 2;
                y = yCell * y1 + yCell - 2;
                frm.gh.drawLine(x, y, x + 10, y);
                frm.gh.drawLine(x, y, x, y - 10);
            }
            else { //hiện mũi tên góc trên trái
                x = xCell * x1 + 2;
                y = yCell * y1 + 2;
                frm.gh.drawLine(x, y, x + 10, y);
                frm.gh.drawLine(x, y, x, y + 10);
            }
            //giả lập thực hiện công việc của thread
            MySleep(tgchay);
            //xác định vị trí mới của thread
            HieuchinhVitri();
            x2 = Pos.x; y2 = Pos.y;
            //thử xin khóa truy xuất cell mới (x2,y2)
            while (true) {
                kq = frm.mutList[y2][x2].tryAcquire(30, TimeUnit.MILLISECONDS);
                if (kq == true || fstart == false) break;
            }
            //Xóa icon ở vị trí cũ
            frm.gh.drawImage(frm.blackPic,xCell * x1, yCell * y1,null);
            //mở khóa cell cũ (x1,y1) cho các thread khác truy xuất
            frm.mutList[y1][x1].release();
            if (kq == false && fstart == false) { //xóa thread
                this.stop();
                this.fstop = true;
                return;
            }
        }
    }
    catch (Exception e) { this.stop(); }
    //xóa icon của thread trước khi ngừng
    x1 = Pos.x; y1 = Pos.y;
    frm.gh.drawImage(frm.blackPic,xCell * x1, yCell * y1,null);
    //mở khóa cell (x1,y1) cho các thread khác truy xuất
    frm.mutList[y1][x1].release();
```

```
        //dừng Thread
        fstop = true;
        stop();
}

  //================================================
  //Hiệu chỉnh góc chạy của thread
  //để tránh các trường hợp thread chạy thẳng đứng hay ngang
  //================================================
  double ChinhGocChay(double dblGocChay)  {
        double goc = dblGocChay;
        if (0 <= goc && goc < 90) return 45;
        if (90 <= goc && goc < 180) return 135;
        if (180 <= goc && goc < 270) return 225;
        if (270 <= goc) return 315;
        return goc;
  }
  //============================================================
  //Tính góc phản xạ mới khi thread đụng thành đứng (bên trái hay phải).
  //============================================================
  double DoiGocChayX(double dblGocChay) {
        double goc;
        if (dblGocChay > 0 && dblGocChay < 180) goc = 180 - dblGocChay;
        else goc = 180 + 360 - dblGocChay;
        return ChinhGocChay(goc);
  }
  //============================================================
  //Tính góc phản xạ mới khi thread đụng thành ngang (trên hay dưới).
  //============================================================
  double DoiGocChayY(double dblGocChay) {
        return ChinhGocChay(360 - dblGocChay);
  }
  //============================================================
  //Hiệu chỉnh vị trí của thread
  //============================================================
  public void HieuchinhVitri() {
        int x, y;
        x = Pos.x;
        y = Pos.y;
        if (x == 0 || x == xCount - 1 || y == 0 || y == yCount - 1) {
           //icon đụng thành ngang hay dọc -> thay đổi góc chạy
           if (x == 0 || x == xCount - 1)  dblGocChay = DoiGocChayX(dblGocChay);
           else if (y == 0 || y == yCount - 1) dblGocChay = DoiGocChayY(dblGocChay);
        }
        //Hiệu chỉnh tọa độ x của thread
        tx = 2 * Math.cos(dblGocChay * PI / 180);
        x = x + (int)tx;
        if (x < 0) x = 0;
        else if (x >= xCount) x = xCount - 1;
        //Hiệu chỉnh tọa độ y của thread
```

```
            ty = 2 * Math.sin(dblGocChay * PI / 180);
            y = y + (int)ty;
            if (y < 0) y = 0;
            else if (y >= yCount)  y = yCount - 1;
            //chỉnh góc chạy khi đụng 1 trong 4 góc
            if (x == 0 && y == 0) //góc trên trái
               ChinhGocChay(dblGocChay + 45);
            else if (x == 0 && y == yCount - 1) //góc dưới trái
               ChinhGocChay(dblGocChay + 45);
            else if (x == xCount - 1 && y == 0) //góc trên phải
               ChinhGocChay(dblGocChay + 45);
            else if (x == xCount - 1 && y == yCount - 1) //góc dưới phải
               ChinhGocChay(dblGocChay + 45);
            //Lưu vị trí mới
            Pos.x = (int)x;
            Pos.y = (int)y;
        }
}
```

4. Làm hiển thị cửa sổ soạn mã nguồn của class MainForm, dời cursor đến đầu các vị trí được tô màu dưới đây rồi thêm code được tô màu vào như sau :

```
package nbthreaddemo2;
//import các package cần dùng
import java.util.*;
import java.awt.*;
import java.net.*;
import javax.imageio.*;
import javax.swing.*;
import java.util.concurrent.*;

//định nghĩa form ứng dụng
public class MainForm extends javax.swing.JFrame {
      //định nghĩa các thuộc tính cần dùng
      public Semaphore[][] mutList;
      public Image blackPic;
      MyThread[] threadLst;
      final int xCell = 30;
      final int yCell = 30;
      final int xCount = 25;
      final int yCount = 20;
      Graphics gh;
     //tạo đối tượng sinh số ngẫu nhiên
     public Random rnd = new Random();
     //tác vụ khởi tạo form
     public MainForm() {
          setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
          //thiết lập tác vụ xử lý sự kiện ấn phím trên form
          addKeyListener(new java.awt.event.KeyAdapter() {
          public void keyPressed(java.awt.event.KeyEvent evt) {
             formKeyPressed(evt);
          }
```

```
        });
        //thiết lập lại kích thước form theo yêu cầu
        this.setSize(xCount * 30, yCount * 30);
        this.setLocation(0, 0);
        //thiết lập màu nền đen cho form
        this.getContentPane().setBackground(Color.BLACK);
        //đọc bitmap miêu tả cell nền từ file
        URL url;
        try {
        url = this.getClass().getClassLoader().getResource
             ("images/Black.jpg");
        blackPic = ImageIO.read(url);
        } catch (Exception e) {}
        //tạo danh sách chứa 26 thread từ A-Z
        threadLst = new MyThread[26];
        int i;
        //tạo ma trận semaphore nhị phân để bảo vệ các cell màn hình
        mutList = new Semaphore[yCount][xCount];
        int h, cot=0;
        for (h = 0; h < yCount; h++)
           for (cot = 0; cot < xCount; cot++)
              mutList[h][cot] = new Semaphore(1);
        //Lặp thiết lập trạng thái ban đầu cho 26 thread từ A-Z
        for (i = 0; i < 26; i++) {
           threadLst[i] = new MyThread(rnd, xCount, yCount,this);
           threadLst[i].fstop = threadLst[i].fsuspend = threadLst[i].fstart = false;
           char c = (char)(i + 65);
           try { //đọc bitmap miêu tả thread c từ file
             url = this.getClass().getClassLoader().getResource
                    ("images/Image" + c + ".jpg");
             threadLst[i].Pic = ImageIO.read(url);
           } catch (Exception e) {
              System.out.println(e.toString());
           }
        }
    }
    //tác vụ xử lý việc ấn phím
    private void formKeyPressed(java.awt.event.KeyEvent evt) {
        //lưu đối tượng Graphics của form để vẽ icon khi cần
        gh = this.getGraphics();
        //xác định mã phím ấn, nếu không phải từ A-Z thì phớt lờ
        int newch = evt.getKeyCode();
        if (newch < 0x41 || newch > 0x5a) return;
        //xác định chức năng mà user muốn và thực hiện
        if (evt.isControlDown() && evt.isShiftDown()) { //dừng Thread
           threadLst[newch - 65].fstart = false;
        }
        else if (evt.isControlDown()) { //giảm độ ưu tiên tối thiểu
           threadLst[newch - 65].setPriority(1);
        }
```

```
            else if (evt.isControlDown() && evt.isAltDown()) { //tạm dừng thread
               if (threadLst[newch - 65].fstart && !threadLst[newch - 65].fsuspend)
               {
                  threadLst[newch - 65].suspend();
                  threadLst[newch - 65].fsuspend = true;
               }
            }
            else if (evt.isAltDown()) { //cho thread chạy lại
               if (threadLst[newch - 65].fstart && threadLst[newch - 65].fsuspend)
               {
                  threadLst[newch - 65].resume();
                  threadLst[newch - 65].fsuspend = false;
               }
            }
            else if (evt.isShiftDown()) { //tăng độ ưu tiên tối đa
                 threadLst[newch - 65].setPriority(31);
            }
            else {  //tạo mới thread và bắt đầu chạy
               if (!threadLst[newch - 65].fstart) {
                  threadLst[newch - 65].fstart = true;
                  threadLst[newch - 65].fsuspend = false;
                  threadLst[newch - 65].start();
               }
            }
      }
   }
```

5. Chọn menu Run.Run Project để dịch và chạy thử chương trình. Nếu có lỗi từ vựng và cú pháp thì sữa, nếu có lỗi run-time thì debug (thông qua các chức năng trong menu Debug) để xác định lỗi rồi sữa lỗi.
5. Nếu chương trình hết lỗi, cửa sổ chương trình sẽ hiển thị với cửa sổ có nền đen.
6. Hãy tạo thử từng thread (gỏ phím chữ tương ứng) và quan sát quỹ đạo chạy và tốc độ chạy của chúng. Quan sát hiện tượng đè nhau giữa các thread còn không.
7. Quan sát hiện tượng deadlock, nếu có thử phân tích nguyên nhân rồi giết 1 hay nhiều thread gây deadlock để các thread còn lại chạy tiếp.