ชื่อ-นามสกุล นายณัฐเขษม ทรัพย์สิน รหัสประจำตัวนักศึกษา 6604062630151 *ปีการศึกษา 2567*

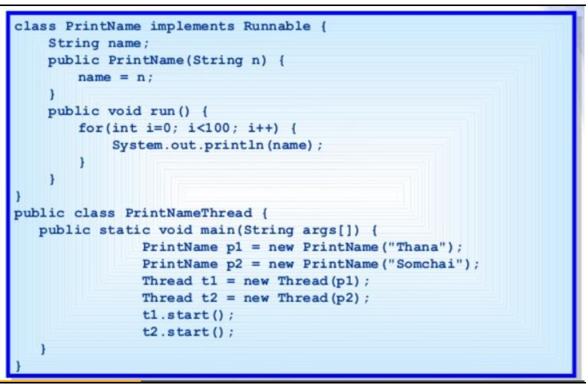
แบบฝึกหัดปฏิบัติการคาบที่ 11: Thread

คำสั่ง ให้ศึกษาหลักการของเธรดต่อไปนี้

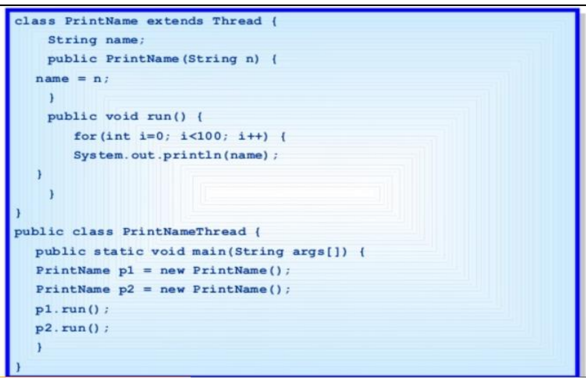
1 เธรด จะแตกต่างจาก Process ที่ทำงานภายใต้ระบบปฏิบัติการแบบ multi-tasking ตรงที่ process แต่ละ process จะมีความเป็นอิสระจากกัน แต่เธรดแต่ละเธรดอาจใช้ข้อมูลร่วมกัน คลาสแบบเธรดในภาษาจาวาคือคลาสที่สืบทอดมาจากคลาสที่ชื่อ Thread หรือคลาสที่ implements อินเตอร์เฟสชื่อ Runable ภายในคลาสแบบเธรดจะต้องมีเมธอด run() ที่ไม่มีการรับอาร์กูเม้นต์ใด ๆ เข้ามา สถานะของ เธรดอาจจะเป็น New runnable running blocked หรือ dead

วงจรชีวิตของแต่ละเธรดเริ่มต้นที่ขั้น new เมื่อถูกเริ่มสร้างและจะอยู่ที่ขั้นนี้จนกระทั่งเมธอด start ถูกเรียกประมวลผล ซึ่งจะทำให้เธรดนั้นๆ อยู่ที่ขั้น ready โดยที่เธรดที่มี priority สูงที่สุดจะถูก ทำงานก่อนโดยเข้าสู่ขั้น running แต่ละเธรดจะเข้าสู่ขั้นสุดท้ายของวงจรชีวิตคือ dead เมื่อเธรดนั้น ทำงานในเมธอด run เสร็จเรียบร้อยหรือเมื่อถูกหยุดการทำงานด้วยเหตุผลใดๆ ก็ตาม ออปเจ็คแบบเธรดเมื่อลงทะเบียนไว้กับตัวตารางเวลาแล้ว อาจยังไม่มีการรันโปรแกรมโดยทันที แต่ทั้งนี้จะขึ้นอยู่กับสถานะของออปเจ็ค

ตัวอย่าง

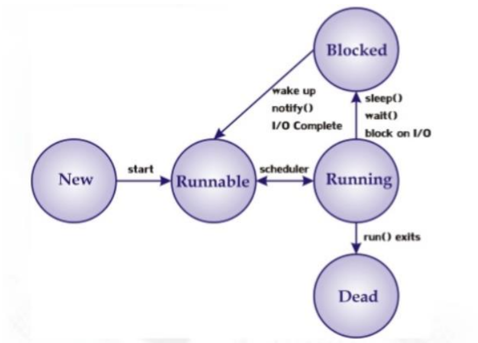


```
class PrintName implements Runnable {
    String name;
    public PrintName(String n) {
        name = n;
    }
    public void run() {
        for(int i=0; i<100; i++) {
            System.out.println(name);
        }
    }
}
public class PrintNameThread {
    public static void main(String args[]) {
        PrintName p1 = new PrintName("Thana");
        PrintName p2 = new PrintName("Somchai");
        Thread t1 = new Thread(p1);
        Thread t2 = new Thread(p2);
        t1.start();
        t2.start();
    }
}
```



```
class PrintName extends Thread {
    String name;
    public PrintName(String n) {
    name = n;
    }
    public void run() {
        for(int i=0; i<100; i++) {
        System.out.println(name);
    }
    }
}
public class PrintNameThread {
    public static void main(String args[]) {
    PrintName p1 = new PrintName();
    PrintName p2 = new PrintName();
    p1.run();
    p2.run();
    }
}
```

วงจรการทำงานของเธรด



2.จงอธิบายผลลัพธ์ของการทำงานโดยการใช้ Thread ต่อไปนี้

```
class Thread1 extends Thread{
    Thread1(String name){
        super(name);
    }
    public void run(){
        for(int i=0;i<10;i++){
            System.out.println(getName()+" ");
        }
    }
}
class TestThread1{
    public static void main(String args[]){
        new Thread1("A").start();
        new Thread1("B").start();
    }
}
```

getName ที่ตั้งไว้เป็น "A" , "B"

แต่ละ Thread ทำงานแยกกัน โดย Thread A เริ่มก่อน ปริ้นทั้งหมด 10 ครั้ง และ Thread B ทำงานต่ออีก 10 ครั้ง

3.ให้ศึกษาตัวอย่างของการใช้ Thread แทน Timer ต่อไปนี้

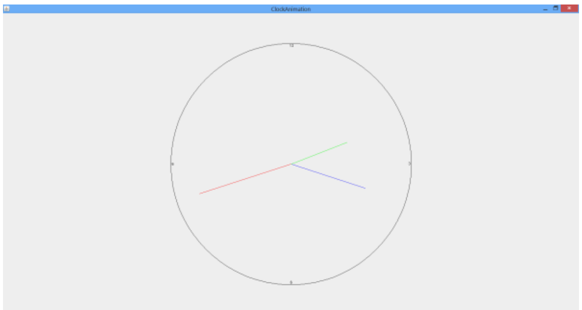
```
 1 import java.awt.*;
 2 import java.util.Formatter;
 3 import javax.swing.*;
 9 public class BouncingBallSimple extends JPanel {
10    // Container box's width and height
11    private static final int BOX_WIDTH = 640;
12    private static final int BOX_HEIGHT = 480;
13
14    // Ball's properties
15    private float ballRadius = 200; // Ball's radius
16    private float ballX = ballRadius + 50; // Ball's center (x, y)
17    private float ballY = ballRadius + 20;
18    private float ballSpeedX = 3;   // Ball's speed for x and y
19    private float ballSpeedY = 2;
20
21    private static final int UPDATE_RATE = 30; // Number of refresh per second
22
23    /** Constructor to create the UI components and init game objects. */
24    public BouncingBallSimple() {
25       this.setPreferredSize(new Dimension(BOX_WIDTH, BOX_HEIGHT));
26
27       // Start the ball bouncing (in its own thread)
28       Thread gameThread = new Thread() {
29          public void run() {
30             while (true) { // Execute one update step
31                // Calculate the ball's new position
32                ballX += ballSpeedX;
33                ballY += ballSpeedY;
34                // Check if the ball moves over the bounds
35                // If so, adjust the position and speed.
36                if (ballX - ballRadius < 0) {
37                   ballSpeedX = -ballSpeedX; // Reflect along normal
38                   ballX = ballRadius; // Re-position the ball at the edge
39                } else if (ballX + ballRadius > BOX_WIDTH) {
40                   ballSpeedX = -ballSpeedX;
41                   ballX = BOX_WIDTH - ballRadius;
42                }
43                // May cross both x and y bounds
44                if (ballY - ballRadius < 0) {
45                   ballSpeedY = -ballSpeedY;
46                   ballY = ballRadius;
47                } else if (ballY + ballRadius > BOX_HEIGHT) {
48                   ballSpeedY = -ballSpeedY;
49                   ballY = BOX_HEIGHT - ballRadius;
50                }
51                // Refresh the display
52                repaint(); // Callback paintComponent()
53                // Delay for timing control and give other threads a chance
54                try {
55                   Thread.sleep(1000 / UPDATE_RATE);  // milliseconds
56                } catch (InterruptedException ex) { }
57             }
58          }
59       };
60       gameThread.start();  // Callback run()
61    }
62
63    /** Custom rendering codes for drawing the JPanel */
64    @Override
65    public void paintComponent(Graphics g) {
66       super.paintComponent(g);    // Paint background
67
```

```
68        // Draw the box
69        g.setColor(Color.BLACK);
70        g.fillRect(0, 0, BOX_WIDTH, BOX_HEIGHT);
71
72        // Draw the ball
73        g.setColor(Color.BLUE);
74        g.fillOval((int) (ballX - ballRadius), (int) (ballY - ballRadius),
75              (int)(2 * ballRadius), (int)(2 * ballRadius));
76
77        // Display the ball's information
78        g.setColor(Color.WHITE);
79        g.setFont(new Font("Courier New", Font.PLAIN, 12));
80        StringBuilder sb = new StringBuilder();
81        Formatter formatter = new Formatter(sb);
82        formatter.format("Ball @(%3.0f,%3.0f) Speed=(%2.0f,%2.0f)", ballX, ballY,
83              ballSpeedX, ballSpeedY);
84        g.drawString(sb.toString(), 20, 30);
85     }
86
87     /** main program (entry point) */
88     public static void main(String[] args) {
89        // Run GUI in the Event Dispatcher Thread (EDT) instead of main thread.
90        javax.swing.SwingUtilities.invokeLater(new Runnable() {
91           public void run() {
92              // Set up main window (using Swing's Jframe)
93              JFrame frame = new JFrame("A Bouncing Ball");
94              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
95              frame.setContentPane(new BouncingBallSimple());
96              frame.pack();
97              frame.setVisible(true);
98           }
99        });
100    }
101 }
```

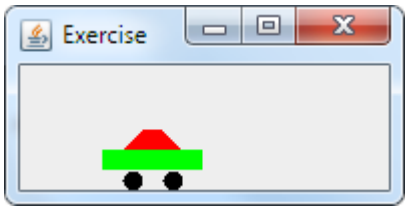
จงอธิบายผลลัพธ์ของการทำงานโดยที่มีการใช้ Thread ยกตัวอย่างพร้อมอธิบายบรรทัดที่มีการใช้

บรรทัด 24 - 57 อัพเดตตำแหน่งลูกบอล โดยเพิ่มความเร็ว ballspeedX และ ballspeedY ลงในตำแหน่ง ballX และ ballY และตรวจสอบการชนขอบ ถ้าชนขอบให้เด้งกลับในทิศทางตรงข้าม และ repaint() แสดงใหม่ และ sleep Thread 1000/update-Rate เพื่อควบคุมอัตราการอัพเดต

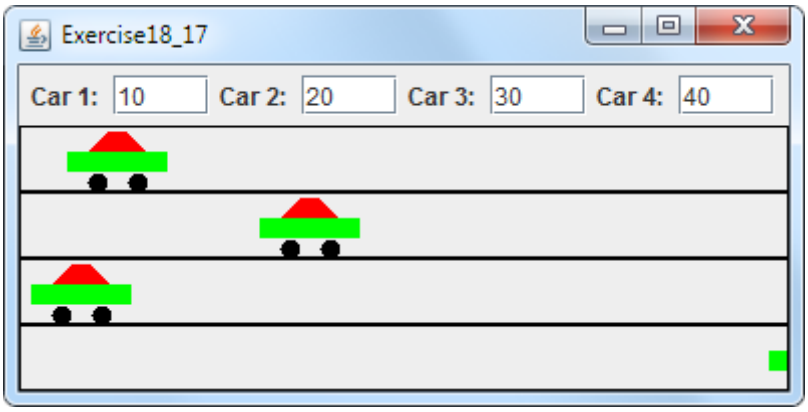
4.ให้เปลี่ยนการทำงานของ Clock Animation จากที่ใช้ Timer ให้เปลี่ยนไปใช้ Thread จากคลาส Thread แทน



5. ให้เปลี่ยนการทำงานของ Race car จากที่ใช้ Timer ให้เปลี่ยนไปใช้ Thread จากคลาส Thread แทน



6. จากคลาส Race car ในปฏิบัติการข้อที่ผ่านมาให้เพิ่มจำนวนรถเป็น 4 คันจำลองให้เป็นสนามแข่งรถ และสามารถกำหนดความเร็วให้รถแต่ละคันได้ จากที่ใช้ Timer ให้เปลี่ยนไปใช้ Thread จากคลาส Thread แทน



7.เขียนโปรแกรมที่แสดงรูปบอลลอนในตำแหน่ง random ใน Panel ให้ใช้ปุ่มลูกศรซ้ายและลูกศรขวาในการเล็งตำแหน่งของปืนให้ตรงกับบอลลูน ใช้ปุ่มลูกศร up-arrow สำหรับยิงลูกกระสุน เมื่อลูกกระสุนถูกบอลลูนให้บอลลูนที่โดนกระสุนหายไป และ random บอลลูนที่ตำแหน่งใหม่ขึ้นมาจากที่ใช้ Timer ให้เปลี่ยนไปใช้ Thread จากคลาส Thread แทน