Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з комп’ютерного практикуму №1 з дисципліни
«Технології паралельних обчислень»

«Розробка потоків та дослідження пріоритету запуску потоків»

Виконав(ла) *ІП-11 Прищепа В. С.* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Дифучин А. Ю.* _____
(прізвище, ім'я, по батькові)

Київ 2024

З**авдання:**

1. Реалізуйте програму імітації руху більярдних кульок, в якій рух кожної кульки відтворюється в окремому потоці (див. презентацію «Створення та запуск потоків в java» та приклад). Спостерігайте роботу програми при збільшенні кількості кульок. Поясніть результати спостереження. Опишіть переваги потокової архітектури програм. 10 балів.
2. Модифікуйте програму так, щоб при потраплянні в «лузу» кульки зникали, а відповідний потік завершував свою роботу. Кількість кульок, яка потрапила в «лузу», має динамічно відображатись у текстовому полі інтерфейсу програми. 10 балів.
3. Виконайте дослідження параметру priority потоку. Для цього модифікуйте програму «Більярдна кулька» так, щоб кульки червоного кольору створювались з вищим пріоритетом потоку, в якому вони виконують рух, ніж кульки синього кольору. Спостерігайте рух червоних та синіх кульок при збільшенні загальної кількості кульок. Проведіть такий експеримент. Створіть багато кульок синього кольору (з низьким пріоритетом) і одну червоного кольору, які починають рух в одному й тому ж самому місці більярдного стола, в одному й тому ж самому напрямку та з однаковою швидкістю. Спостерігайте рух кульки з більшим пріоритетом. Повторіть експеримент кілька разів, значно збільшуючи кожного разу кількість кульок синього кольору. Зробіть висновки про вплив пріоритету потоку на його роботу в залежності від загальної кількості потоків. 20 балів.
4. Побудуйте ілюстрацію методу join() класу Thread через взаємодію потоків, що відтворюють рух більярдних кульок різного кольору. Поясніть результат, який спостерігається. 10 балів.
5. Створіть два потоки, один з яких виводить на консоль символ ‘-‘, а інший – символ ‘|’. Запустіть потоки в основній програмі так, щоб вони виводили свої символи в рядок. Виведіть на консоль 100 таких рядків. Поясніть виведений результат. 10 балів. Використовуючи найпростіші методи управління потоками, добийтесь почергового виведення на консоль символів. 15 балів.

6. Створіть клас Counter з методами increment() та decrement(), які збільшують та зменшують значення лічильника відповідно. Створіть два потоки, один з яких збільшує 100000 разів значення лічильника, а інший – зменшує 100000 разів значення лічильника. Запустіть потоки на одночасне виконання. Спостерігайте останнє значення лічильника. Поясніть результат. 10 балів. Використовуючи синхронізований доступ, добийтесь правильної роботи лічильника при одночасній роботі з ним двох і більше потоків. Опрацюйте використання таких способів синхронізації: 8 синхронізований метод, синхронізований блок, блокування об’єкта. Порівняйте способи синхронізації. 15 балів.

Виконання:

1 Завдання

Лістинг

Ball.java

```
package org.example.task1;

import java.awt.*;
import java.awt.geom.Ellipse2D;
import java.util.Random;

public class Ball {
    private Component canvas;
    private static final int xSIZE = 20;
    private static final int ySIZE = 20;

    private int x = 0;
    private int y = 0;
    private int dx = 2;
    private int dy = 2;

    public Ball(Component c) {
        this.canvas = c;

        if (Math.random() < 0.5) {
            x = new Random().nextInt(this.canvas.getWidth());
            y = 0;
        } else {
            x = 0;
```

```
            y = new Random().nextInt(this.canvas.getHeight());
        }
    }

    public void draw(Graphics2D g2) {
        g2.setColor(Color.black);
        g2.fill(new Ellipse2D.Double(x, y, xSIZE, ySIZE));
    }

    public void move() {
        x += dx;
        y += dy;
        if (x < 0) {
            x = 0;
            dx = -dx;
        }
        if (x + xSIZE >= this.canvas.getWidth()) {
            x = this.canvas.getWidth() - xSIZE;
            dx = -dx;
        }
        if (y < 0) {
            y = 0;
            dy = -dy;
        }
        if (y + ySIZE >= this.canvas.getHeight()) {
            y = this.canvas.getHeight() - ySIZE;
            dy = -dy;
        }
        this.canvas.repaint();
    }

}
```

BallCanvas.java

```
package org.example.task1;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
public class BallCanvas extends JPanel {
    private ArrayList<Ball> balls = new ArrayList<>();

    public void add(Ball b){
        this.balls.add(b);
    }
```

```
    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        for (Ball b:balls) {
            b.draw(g2);
        }
    }

}
```

BallThread.java

```
package org.example.task1;

public class BallThread extends Thread{
    private Ball b;

    public BallThread(Ball ball){
        b =ball;
    }

    @Override
    public void run(){
        try {
            for (int i = 0; i < 100000000; i++) {
                b.move();
                System.out.println("thread name = "+
                        Thread.currentThread().getName());
                Thread.sleep(5);
            }
        } catch (InterruptedException ex){

        }
    }

}
```

Bounce.java

```
package org.example.task1;
import javax.swing.*;

public class Bounce {
    public static void main(String[] args) {
        BounceFrame frame = new BounceFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

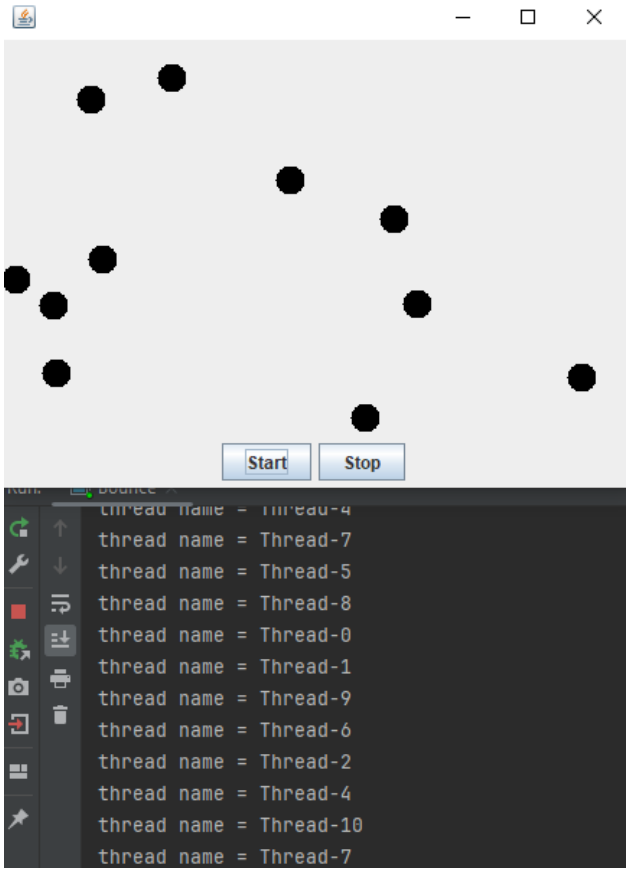
```
        frame.setVisible(true);
        System.out.println("Thread Name = " +
                Thread.currentThread().getName());
    }
}
```

BounceFrame.java

```
package org.example.task1;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class BounceFrame extends JFrame {
    private BallCanvas canvas;
    public static final int WIDTH = 450;
    public static final int HEIGHT = 350;


    public BounceFrame(){
        this.setSize(WIDTH, HEIGHT);
        this.canvas = new BallCanvas();
        System.out.println("In Frame Thread name = "+
                Thread.currentThread().getName());
        Container content = this.getContentPane();
        content.add(this.canvas,BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel();
        JButton buttonStart = new JButton("Start");
        JButton buttonStop = new JButton("Stop");
        buttonStart.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Ball b = new Ball(canvas);
                canvas.add(b);

                BallThread thread = new BallThread(b);
                thread.start();
                System.out.println("Thread name = "+
                        Thread.currentThread().getName());

            }
        });
        buttonStop.addActionListener(new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
        buttonPanel.add(buttonStart);
        buttonPanel.add(buttonStop);

        content.add(buttonPanel, BorderLayout.SOUTH);
    }
}
```

Результат виконання



Програма імітує рух кульок, кожна кулька рухається в окремому потоці, виконання потоків не послідовне.

2 Завдання

Лістинг

Ball.java

```
package org.example.task2;

import java.awt.*;
import java.awt.geom.Ellipse2D;
```

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Ball {
    private Component canvas;
    private static final int xSIZE = 20;
    private static final int ySIZE = 20;

    private int x = 0;
    private int y = 0;
    private int dx = 2;
    private int dy = 2;

    private boolean isInPocket = false;
    private final List<Pocket> pockets;

    public List<Pocket> getPockets(){return pockets;}
    public void setInPocket(boolean isInPocket) {
        this.isInPocket = isInPocket;
    }
    public boolean isInPocket() {return isInPocket;}
    public boolean isBallInPocket() {
        for (Pocket pocket : pockets) {
            if (pocket.isBallInPocket(x, y)) {
                return true;
            }
        }
        return false;
    }
    public Ball(ArrayList<Pocket> pockets, Component c) {
        this.canvas = c;
        this.pockets = pockets;
        y = 0;
        if (Math.random() < 0.5) {
            x = new Random().nextInt(this.canvas.getWidth());
            y = 0;
        } else {
            x = 0;
            y = new Random().nextInt(this.canvas.getHeight());
        }
    }

    public void draw(Graphics2D g2) {
        g2.setColor(Color.blue);
```

```
        g2.fill(new Ellipse2D.Double(x, y, xSIZE, ySIZE));
    }

    public void move() {
        x += dx;
        y += dy;

        if (isBallInPocket()) {
            isInPocket = true;
            System.err.println("Ball into a pocket!");
            this.canvas.repaint();
            return;
        }

        if (x < 0) {
            x = 0;
            dx = -dx;
        }
        if (x + xSIZE >= this.canvas.getWidth()) {
            x = this.canvas.getWidth() - xSIZE;
            dx = -dx;
        }
        if (y < 0) {
            y = 0;
            dy = -dy;
        }
        if (y + ySIZE >= this.canvas.getHeight()) {
            y = this.canvas.getHeight() - ySIZE;
            dy = -dy;
        }
        this.canvas.repaint();
    }

}
```

BallCanvas.java

```
package org.example.task2;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
public class BallCanvas extends JPanel {
    private ArrayList<Ball> balls = new ArrayList<>();
    private ArrayList<Pocket> pockets;
    private int inPocketCounter=0;
```

```
    JLabel inPocketCounterLabel;
    public void add(Ball b){
        this.balls.add(b);
    }
    public ArrayList<Pocket> getPockets() {
        return pockets;
    }
    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

        for (Ball ball : balls) {
            if (ball.isInPocket()) {
                inPocketCounter++;
            }
        }
        pockets = generatePockets();
        paintPockets(g2);
        inPocketCounterLabel.setText("Amount in pocket: " + inPocketCounter);
        balls.removeIf(ball -> ball.isInPocket());

        for (Ball b:balls) {
            b.draw(g2);
        }
    }
    private  ArrayList<Pocket> generatePockets(){
        ArrayList<Pocket> pockets = new ArrayList<>();
        pockets.add(new Pocket(0, 0, 20));
        pockets.add(new Pocket(0, this.getHeight() - 40, 20));
        pockets.add(new Pocket(this.getWidth() - 40, 0, 20));
        pockets.add(new Pocket(this.getWidth() - 40, this.getHeight() - 40, 20));

        return pockets;
    }
    private void paintPockets(Graphics2D g2){
        for (Pocket p: pockets) {
            p.draw(g2);
        }
    }

}
```

BallThread.java

```java
package org.example.task2;

public class BallThread extends Thread{
    private Ball b;

    public BallThread(Ball ball){
        b =ball;
    }

    @Override
    public void run(){
        try {
            for (int i = 0; i < 100000000; i++) {
                if (b.isInPocket()) {
                    this.interrupt();
                    return;
                }
                b.move();
                System.out.println("thread name = "+
                        Thread.currentThread().getName());
                Thread.sleep(5);
            }
        } catch (InterruptedException ex){

        }
    }

}
```

Bounce.java

```java
package org.example.task2;

import javax.swing.*;

public class Bounce {
    public static void main(String[] args) {
        BounceFrame frame = new BounceFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setVisible(true);
        System.out.println("Thread Name = " +
                Thread.currentThread().getName());
    }
}
```

BounceFrame.java

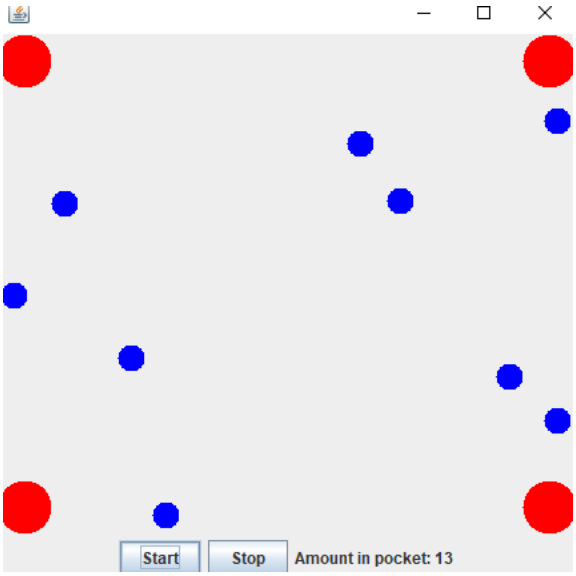
```java
package org.example.task2;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class BounceFrame extends JFrame {
    private BallCanvas canvas;
    public static final int WIDTH = 450;
    public static final int HEIGHT = 450;
    public  JLabel inPocketCountLabel;

    public BounceFrame(){
        this.setSize(WIDTH, HEIGHT);
        this.canvas = new BallCanvas();
        System.out.println("In Frame Thread name = "+
                Thread.currentThread().getName());
        Container content = this.getContentPane();
        content.add(this.canvas,BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel();
        JButton buttonStart = new JButton("Start");
        JButton buttonStop = new JButton("Stop");

        inPocketCountLabel = new JLabel("Amount in pocket: 0");
        canvas.inPocketCounterLabel = inPocketCountLabel;

        buttonStart.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Ball b = new Ball(canvas.getPockets(),canvas);
                canvas.add(b);

                BallThread thread = new BallThread(b);
                thread.start();
                System.out.println("Thread name = "+
                        Thread.currentThread().getName());

            }
        });
        buttonStop.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
```

```
        });

        buttonPanel.add(buttonStart);
        buttonPanel.add(buttonStop);
        buttonPanel.add(inPocketCountLabel);

        content.add(buttonPanel, BorderLayout.SOUTH);
    }
}
```

Pocket.java

```
package org.example.task2;

import java.awt.*;
import java.awt.geom.Ellipse2D;

public class Pocket {
    private final int x;
    private final int y;
    private int radius;

    public int getX(){return x;}
    public int getY(){return y;}
    public int getRadius(){return radius;}
    public Pocket(int x,int y,int radius){
        this.x=x;
        this.y=y;
        this.radius=radius;
    }
    public void draw(Graphics2D g2){
        g2.setColor(Color.RED);
        g2.fill(new Ellipse2D.Double(x,y,radius*2,radius*2));
    }
    public boolean isBallInPocket(int x,int y){
        double pocketRadius = this.getRadius()*2;
        double distance = Math.sqrt(Math.pow(this.getX()-x,2)+Math.pow(this.getY()-
y,2));
        return distance<=pocketRadius;
    }
}
```

Результат виконання



При потраплянні в лузу кульки зникають і відповідний потік завершує свою роботу, а кількість цих кульок відображається внизу інтерфейса.

3 Завдання

Лістинг

Ball.java

```
package org.example.task3;

import java.awt.*;
import java.awt.geom.Ellipse2D;

public class Ball {
    private Component canvas;
    private static final int xSIZE = 20;
    private static final int ySIZE = 20;

    private int x = 0;
    private int y = 0;
    private int dx = 2;
    private int dy = 2;
    private Color color;
    public Ball( Component c) {
        this.canvas = c;
        x = this.canvas.getWidth()/2;
        y = this.canvas.getHeight()/2;
    }

    public void draw(Graphics2D g2) {
```

```
        if (color== Color.BLUE){g2.setColor(Color.blue);}
        if (color== Color.RED){g2.setColor(Color.red);}
        g2.fill(new Ellipse2D.Double(x, y, xSIZE, ySIZE));
    }

    public void move() {
        x += dx;
        y += dy;

        if (x < 0) {
            x = 0;
            dx = -dx;
        }
        if (x + xSIZE >= this.canvas.getWidth()) {
            x = this.canvas.getWidth() - xSIZE;
            dx = -dx;
        }
        if (y < 0) {
            y = 0;
            dy = -dy;
        }
        if (y + ySIZE >= this.canvas.getHeight()) {
            y = this.canvas.getHeight() - ySIZE;
            dy = -dy;
        }
        this.canvas.repaint();
    }
    public void setColor(Color color) {
        this.color = color;
    }

}
```

BallCanvas.java

```
package org.example.task3;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
public class BallCanvas extends JPanel {
    private ArrayList<Ball> balls = new ArrayList<>();
    public void add(Ball b){
        this.balls.add(b);
    }
```

```
    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

        for (Ball b:balls) {
            b.draw(g2);
        }
    }
}
```

BallThread.java

```
package org.example.task3;

public class BallThread extends Thread{
    private Ball b;

    public BallThread(Ball ball,int priority){
        b =ball;
        this.setPriority(priority);
    }

    @Override
    public void run(){
        try {
            for (int i = 0; i < 100000000; i++) {
                b.move();
                System.out.println("thread name = "+
                        Thread.currentThread().getName());
                Thread.sleep(5);
            }
        } catch (InterruptedException ex){

        }
    }

}
```

Bounce.java

```
package org.example.task3;

import javax.swing.*;

public class Bounce {
    public static void main(String[] args) {
        BounceFrame frame = new BounceFrame();
```

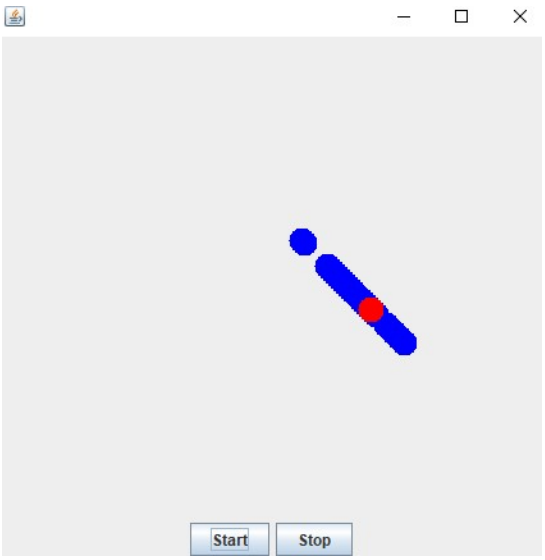
```
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setVisible(true);
        System.out.println("Thread Name = " +
                Thread.currentThread().getName());
    }
}
```

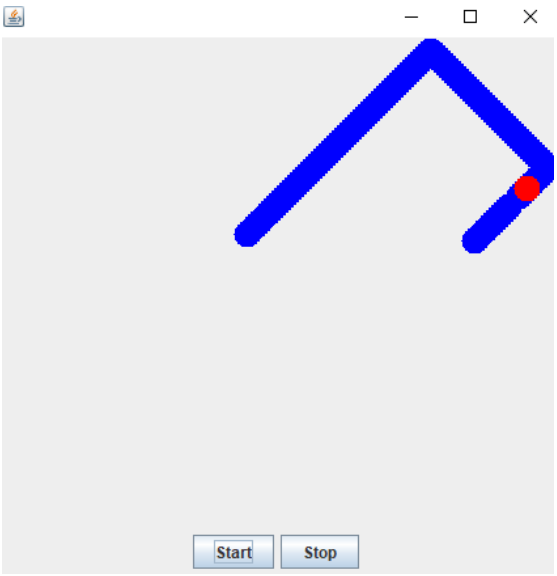
BounceFrame.java

```
package org.example.task3;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class BounceFrame extends JFrame {
    private BallCanvas canvas;
    public static final int WIDTH = 450;
    public static final int HEIGHT = 450;
    private static final int BLUE_PRIORITY = 1;
    private static final int RED_PRIORITY = 10;
    public BounceFrame(){
        this.setSize(WIDTH, HEIGHT);
        this.canvas = new BallCanvas();
        System.out.println("In Frame Thread name = "+
                Thread.currentThread().getName());
        Container content = this.getContentPane();
        content.add(this.canvas,BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel();
        JButton buttonStart = new JButton("Start");
        JButton buttonStop = new JButton("Stop");

        buttonStart.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                int blue_ball_count = 1000;

                for (int i = 0; i < blue_ball_count; i++) {
                    Ball blue_b = new Ball(canvas);
                    blue_b.setColor(Color.BLUE);
                    canvas.add(blue_b)
                    ;
                    BallThread blue_thread = new BallThread(blue_b,BLUE_PRIORITY);
                    blue_thread.start();
```

```
                System.out.println("Thread name = "+
                        Thread.currentThread().getName());
            }
            Ball red_b = new Ball(canvas);
            red_b.setColor(Color.RED);
            canvas.add(red_b);

            BallThread red_thread = new BallThread(red_b, RED_PRIORITY);
            red_thread.start();
            System.err.println(" RED Thread name = " +
                    red_thread.getName());

        }
    });

    buttonStop.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            System.exit(0);
        }
    });

    buttonPanel.add(buttonStart);
    buttonPanel.add(buttonStop);

    content.add(buttonPanel, BorderLayout.SOUTH);
  }
}
```

Результат виконання





На першому малюнку 50 синіх кульок, на другому — 1000. Червона кулька створюється в кінці змійки і намагається видертись у перед. Чим менше кульок, тим простіше червоній видертись у голову змійки.

4 Завдання

Лістинг

Ball.java

```
package org.example.task4;

import java.awt.*;
import java.awt.geom.Ellipse2D;
import java.util.*;

public class Ball {
    private Component canvas;
    private static final int xSIZE = 20;
    private static final int ySIZE = 20;

    private int x = 0;
    private int y = 0;
    private int dx = 2;
    private int dy = 2;
    private Color color;
    private boolean isStopped = false;
    public void stopBall() {
        isStopped = true;
    }
```

```
public void runBall() {
    isStopped = false;
}
public Ball( Component c) {
    this.canvas = c;
    if (Math.random() < 0.5) {
        x = new Random().nextInt(this.canvas.getWidth());
        y = 0;
    } else {
        x = 0;
        y = new Random().nextInt(this.canvas.getHeight());
    }
}

public void draw(Graphics2D g2) {
    if (color== Color.BLUE){g2.setColor(Color.blue);}
    if (color== Color.RED){g2.setColor(Color.red);}
    g2.fill(new Ellipse2D.Double(x, y, xSIZE, ySIZE));
}

public void move() {
    if(isStopped) return;
    x += dx;
    y += dy;

    if (x < 0) {
        x = 0;
        dx = -dx;
    }
    if (x + xSIZE >= this.canvas.getWidth()) {
        x = this.canvas.getWidth() - xSIZE;
        dx = -dx;
    }
    if (y < 0) {
        y = 0;
        dy = -dy;
    }
    if (y + ySIZE >= this.canvas.getHeight()) {
        y = this.canvas.getHeight() - ySIZE;
        dy = -dy;
    }
    this.canvas.repaint();
}
public boolean isStopped() {
    return isStopped;
```

```
    }

    public void setColor(Color color) {
        this.color = color;
    }

}
```

BallCanvas.java

```
package org.example.task4;

import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;
public class BallCanvas extends JPanel {
    private ArrayList<Ball> balls = new ArrayList<>();
    public void add(Ball b){
        this.balls.add(b);
    }

    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;

        for (Ball b:balls) {
            b.draw(g2);
        }
    }
    public void stopBalls() {
        for (Ball ball : balls) {
            ball.stopBall();
        }
    }
    public void runBalls() {
        for (Ball ball : balls) {
            ball.runBall();
        }
    }
}
```

BallThread.java

```
package org.example.task4;

public class BallThread extends Thread {
    private Ball b;
```

```
    public BallThread(Ball ball) {
        b = ball;
    }

    @Override
    public void run() {
        try {
            for (int i = 0; i < 250; i++) {
                if (b.isStopped()) {
                    while (b.isStopped()) {
                        Thread.sleep(0);
                    }
                }
                b.move();
                System.out.println("thread name = " +
                        Thread.currentThread().getName());
                Thread.sleep(5);
            }
        } catch (InterruptedException ex) {

        }
    }

}
```

Bounce.java

```
package org.example.task4;

import javax.swing.*;

public class Bounce {
    public static void main(String[] args) {
        BounceFrame frame = new BounceFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setVisible(true);
        System.out.println("Thread Name = " +
                Thread.currentThread().getName());
    }
}
```

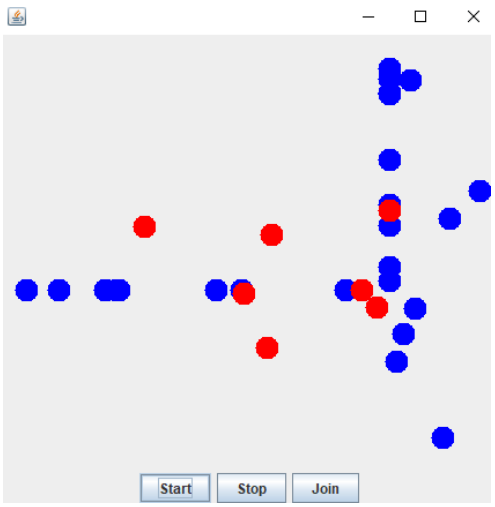
BounceFrame.java

```
package org.example.task4;

import javax.swing.*;
```

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class BounceFrame extends JFrame {
    private BallCanvas canvas;
    public static final int WIDTH = 450;
    public static final int HEIGHT = 450;

    public BounceFrame() {
        this.setSize(WIDTH, HEIGHT);
        this.canvas = new BallCanvas();
        System.out.println("In Frame Thread name = " +
                Thread.currentThread().getName());
        Container content = this.getContentPane();
        content.add(this.canvas, BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel();
        JButton buttonStart = new JButton("Start");
        JButton buttonStop = new JButton("Stop");
        JButton buttonJoin = new JButton("Join");

        buttonStart.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Ball blue_b = new Ball(canvas);
                blue_b.setColor(Color.BLUE);
                canvas.add(blue_b);

                BallThread blue_thread = new BallThread(blue_b);
                blue_thread.start();
                System.out.println("Thread name = " +
                        Thread.currentThread().getName());
            }
        });

        buttonStop.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });

        buttonJoin.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
            canvas.stopBalls();
            Ball b = new Ball(canvas);
            b.setColor(Color.RED);
            canvas.add(b);

            BallThread thread = new BallThread(b);
            thread.start();
            Thread a =new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        thread.join();
                    }catch (InterruptedException ex){
                        ex.printStackTrace();
                    }
                    canvas.runBalls();
                }
            });
            a.start();
        }
    });


    buttonPanel.add(buttonStart);
    buttonPanel.add(buttonStop);
    buttonPanel.add(buttonJoin);

    content.add(buttonPanel, BorderLayout.SOUTH);
  }
}
```

Результат виконання



Після натискання Start створюється синя кулька, а після Join – червона і усі попередні кульки, не залежно від кольору, засинають, очікуючи завершення дії потоку останньої червоної кульки. Після цього усі кульки просинаються і продовжують рухатися аж до завершення дії потоку.

5 Завдання

Лістинг

Improved.java

```
package org.example.task5;

public class Improved extends Thread{
    private final String symbol;
    public Improved(String symbol) {
        this.symbol = symbol;
    }
    private static final Object lock = new Object();

    public void run() {
        for (int i = 0; i < 100; i++) {
            for (int j = 0; j < 100; j++) {
                synchronized (lock) {
                    lock.notify();
                    System.out.print(this.symbol);
                    try {
                        lock.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
            System.out.println();
        }
        System.exit(0);
    }
}
```

Main.java

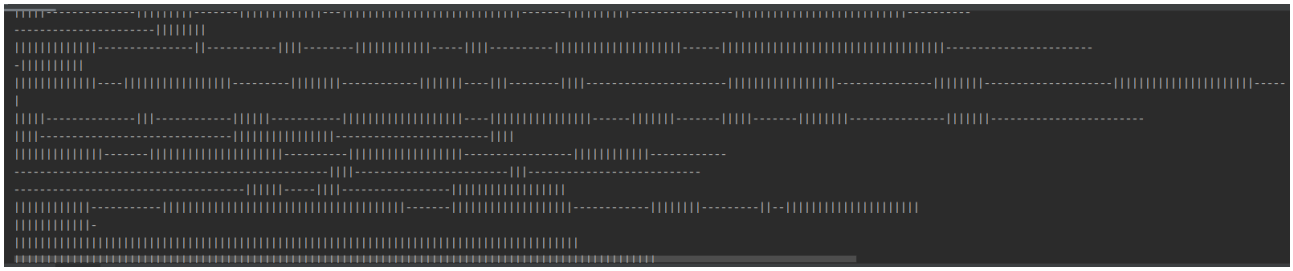
```
package org.example.task5;

public class Main {
    public static void main(String[] args) {
//        Ordinary
//        Improved
```

```
        Improved thread1 = new Improved("-");
        Improved thread2 = new Improved("|");
        thread1.start();
        thread2.start();

    }
}
```

Ordinary.java

```
package org.example.task5;

public class Ordinary  extends Thread{
    private final String symbol;
    public Ordinary(String symbol) {
        this.symbol = symbol;
    }
    public void run() {
        for (int i = 0; i < 100; i++) {
            for (int j = 0; j < 100; j++) {
                System.out.print(this.symbol);
            }
            System.out.println();
        }
    }


}
```

Результат виконання



В обох випадках два потоки виводять — та |, але у першому випадку символи виводяться не почергово, а в другому — почергово завдяки методам notify та wait.

6 Завдання

Лістинг

Counter.java

```
package org.example.task6;

public class Counter {
    private int value = 0;
    private final Object lock = new Object();
    public synchronized void inc() {
        value++;
    }

    public synchronized void dec() {
        value--;
    }
    public void incSyncBlock() {
        synchronized (lock) { value++; }
    }
    public void decSyncBlock() {
        synchronized (lock) { value--; }
    }
    public synchronized int getValue() {
        return value;
    }
    public void unsyncInc() {
        value++;
    }
    public void unsyncDec() {
        value--;
    }
}
```

Main.java

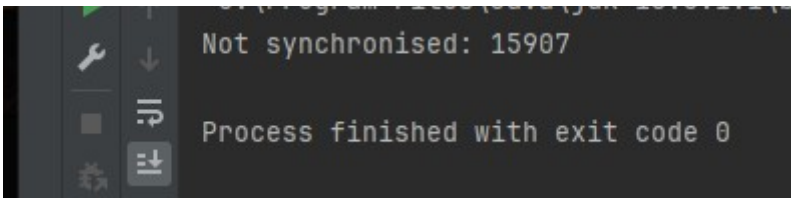
```
package org.example.task6;

import java.util.concurrent.locks.*;
public class Main {
    public static void main(String[] args) {
        Counter counter = new Counter();
```

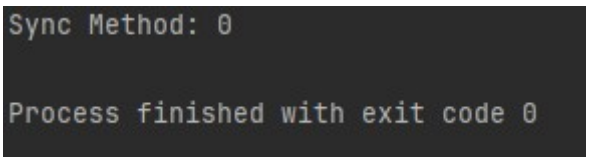
```
        //notSync(counter);
        //syncMethod(counter);
        //syncBlock(counter);
        syncObj(counter);
    }
    private static void notSync(Counter counter) {
        Thread thread1 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 100_000; i++) {
                    counter.unsyncInc();
                }
            }
        });
        Thread thread2 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 100_000; i++) {
                    counter.unsyncDec();
                }
            }
        });
        thread1.start();
        thread2.start();
        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Not synchronised: " + counter.getValue());

    }
    private static void syncMethod(Counter counter) {
        Thread thread1 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 100_000; i++) {
                    counter.inc();
                }
            }
        });
        Thread thread2 = new Thread(new Runnable() {
            @Override
            public void run() {
```

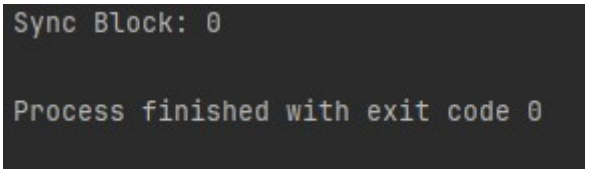
```
            for (int i = 0; i < 100_000; i++) {
                counter.dec();
            }
        }
    });

    thread1.start();
    thread2.start();

    try {
        thread1.join();
        thread2.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("Sync Method: " + counter.getValue());
}
private static void syncBlock(Counter counter) {
    Thread thread1 = new Thread(new Runnable() {
        @Override
        public void run() {
            for (int i = 0; i < 100_000; i++) {
                counter.incSyncBlock();
            }
        }
    });
    Thread thread2 = new Thread(new Runnable() {
        @Override
        public void run() {
            for (int i = 0; i < 100_000; i++) {
                counter.decSyncBlock();
            }
        }
    });

    thread1.start();
    thread2.start();

    try {
        thread1.join();
        thread2.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("Sync Block: " + counter.getValue());
```

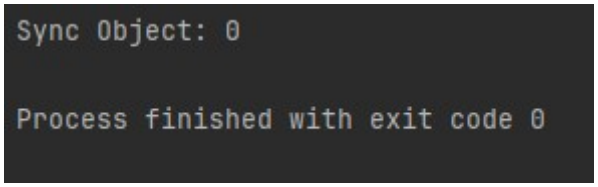
```
    }
    private static void syncObj(Counter counter) {
        final Lock lock = new ReentrantLock();
        Thread thread1 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 100_000; i++) {
                    lock.lock();
                    counter.unsyncInc();
                    lock.unlock();
                }
            }
        });
        Thread thread2 = new Thread(new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i < 100_000; i++) {
                    lock.lock();
                    counter.unsyncDec();
                    lock.unlock();
                }
            }
        });

        thread1.start();
        thread2.start();

        try {
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Sync Object: " + counter.getValue());
    }

}
```

Результат виконання









Два потоки працюють одночасно, один збільшує лічильник 100000 разів, інший — зменшує таку саму кількість разів. Без синхронізованого доступу вийшла дурня, а з використанням синхронізованого методу, синхронізованого блока чи блокування об’єкта лічильник залишився на нулі, що є правильним.

Висновок: Виконуючи лабораторну роботу, я дослідив переваги потокової архітектури програм, параметр priority потоку. Наочно проілюстрував метод join() класу Thread через взаємодію потоків. Освоїв синхронізований доступ, опрацювавши використання таких способів синхронізації: синхронізований метод, синхронізований блок, блокування об’єкта.