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»

Виконала студентка ІП-13 Замковий Дмитро Володимирович
(шифр, прізвище, ім'я, по батькові)

Перевірив ст.викл. Дифучин Антон Юрійович
(прізвище, ім'я, по батькові)

Оцінка _____

Дата оцінювання _____

Київ 2024

Завдання 1

Завдання

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.

Лістинги програми

Файл Main.java

```
import javax.swing.*;


public class Main
{
    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());
    }
}
```

Файл BounceFrame.java

```
import javax.swing.*;
import java.awt.*;


public class BounceFrame extends JFrame
{
    private final BallCanvas canvas;

    public static final int count = 10;

    public static final int WIDTH = 800;
    public static final int HEIGHT = 800;

    public BounceFrame()
    {
        this.setSize(WIDTH, HEIGHT);
        this.setTitle("Bounce program");
        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();
        buttonPanel.setBackground(Color.lightGray);

        JButton buttonStart = new JButton("Start");
        JButton buttonStop = new JButton("Stop");

        buttonStart.addActionListener(e -> {
            for (int i = 0; i < count; i++) {
                Ball b = new Ball(canvas);
                canvas.add(b);

                BallThread thread = new BallThread(b);
                thread.start();

                System.out.println("Thread name = " + thread.getName());
            }
        });

        buttonStop.addActionListener(e -> System.exit(0));

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

Файл BallCanvas.java

```
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;


public class BallCanvas extends JPanel
{
    private final 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);
        }
    }
}
```

Файл BallThread.java

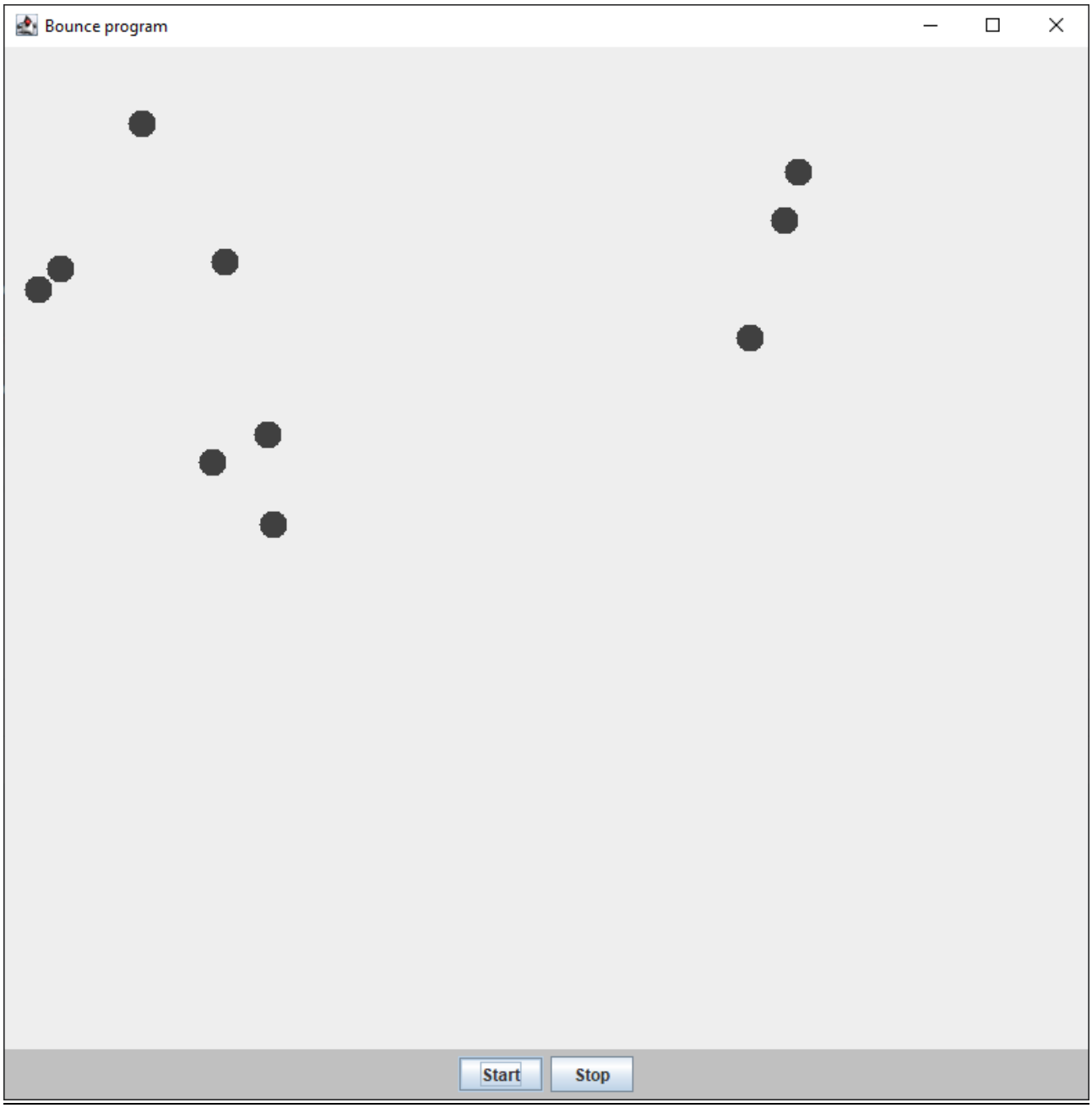
```
public class BallThread extends Thread
{
    private final Ball b;

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

    @Override
    public void run()
    {
        try {
            for (int i = 1; i < 10000; i++){
                b.move();
                System.out.println("Thread name = " +
Thread.currentThread().getName());
                Thread.sleep(5);
            }
        } catch(InterruptedException ignored) {}
    }
}
```

Файл Ball.java

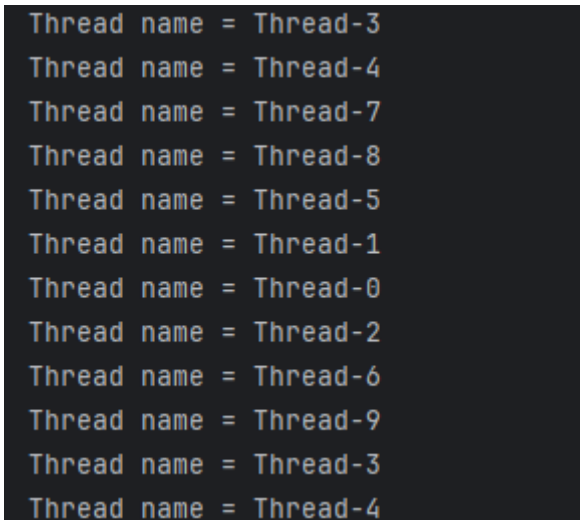
```
import java.awt.*;
import java.awt.geom.Ellipse2D;
import java.util.Random;


public class Ball
{
    private final Component canvas;

    // Розміри кулі
    private static final int XSIZE = 20;
    private static final int YSIZE = 20;

    // Позиція кулі
    private int x;
```

```
    private int y;

    // Вектор кулі
    private int dx = 5;
    private int dy = 5;

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

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

    // Малювання кулі
    public void draw (Graphics2D g2)
    {
        g2.setColor(Color.darkGray);
        g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
    }

    // Переміщення кулі
    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();
    }
}
```

Скріншоти



Інші результати



Висновки

В ході виконання даного завдання було розроблено програму, яка імітує рух більярдних кульок, кожної в окремо потоці. Провівши дослідження я визначив, що фрізи у відображенні з’являють при середньому числі кульок/потоків – близько восьмисот. Перевагами даної архітектури програми можна зазначити легкість розширення функціоналу, що в подальшому використається в наступних завданнях.

Завдання 2

Завдання

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.

Лістинги програми

Файл Main.java

```
import javax.swing.*;


public class Main
{
    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());
    }
}
```

Файл BounceFrame.java

```
import javax.swing.*;
import java.awt.*;


public class BounceFrame extends JFrame
{
    private final ObjectCanvas canvas;

    public static final int count = 10;

    public static final int WIDTH = 1600;
    public static final int HEIGHT = 800;

    public BounceFrame()
    {
        this.setSize(WIDTH, HEIGHT);
        this.setTitle("Bounce program");
        this.canvas = new ObjectCanvas();

        System.out.println("In Frame Thread name = " +
Thread.currentThread().getName());

        Container content = this.getContentPane();
```

```
        content.add(this.canvas, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setBackground(Color.lightGray);

        JButton buttonStart = new JButton("Start");
        JButton buttonStop = new JButton("Stop");
        JLabel labelScoreCounter = new JLabel("Score: " + Score.getScore());

        Score.addListener(() -> {
            labelScoreCounter.setText("Score: " + Score.getScore());
            labelScoreCounter.repaint();
        });



        buttonStart.addActionListener(e -> {
            for (int i = 0; i < count; i++) {
                Ball b = new Ball(canvas);
                canvas.addBall(b);

                BallThread thread = new BallThread(b);
                thread.start();

                System.out.println("Thread name = " + thread.getName());
            }
        });

        buttonStop.addActionListener(e -> System.exit(0));

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

Файл ObjectCanvas.java

```
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;


public class ObjectCanvas extends JPanel
{
    private final ArrayList<Ball> balls = new ArrayList<>();
    private final Pool[] pools = new Pool[6];

    public void addBall(Ball b)
    {
```

```
        this.balls.add(b);
    }

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

        pools[0] = new Pool(0, 0, g2);
        pools[1] = new Pool(super.getWidth() / 2, 0, g2);
        pools[2] = new Pool(super.getWidth(), 0, g2);
        pools[3] = new Pool(0, super.getHeight(), g2);
        pools[4] = new Pool(super.getWidth() / 2, super.getHeight(), g2);
        pools[5] = new Pool(super.getWidth(), super.getHeight(), g2);

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

    public int[][] getPoolsCoordinates()
    {
        int[][] coordinates = new int[pools.length][2];

        for (int i = 0; i < pools.length; i++) {
            coordinates[i][0] = pools[i].getX();
            coordinates[i][1] = pools[i].getY();
        }

        return coordinates;
    }

    public int[] getPoolSize()
    {
        int[] size = new int[2];
        size[0] = Pool.getXSIZE();
        size[1] = Pool.getYSIZE();
        return size;
    }

    public void removeBall(Ball b)
    {
        Score.incrementScore();
        balls.remove(b);
    }
}
```

Файл BallThread.java

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

    // Конструктор класса
    public BallThread(Ball ball)
    {
        b = ball;
    }

    @Override
    public void run()
    {
        try {
            while (!Thread.interrupted()) {
                b.move();
                System.out.println("Thread name = " +
Thread.currentThread().getName());
                if (b.getIsInPool()) {
                    System.out.println("Thread name ended = " +
Thread.currentThread().getName());
                    b.delete();
                    Thread.currentThread().interrupt();
                }
                Thread.sleep(5);
            }
        } catch(InterruptedException ignored) {}
    }
}
```

Файл Ball.java

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


public class Ball
{
    private final Component canvas;

    // Розміри кулі
    private static final int XSIZE = 20;
    private static final int YSIZE = 20;

    // Позиція кулі
    private int x;
    private int y;
```

```
    // Вектор кулі
    private int dx = 5;
    private int dy = 5;

    // Чи була куля забита в лузу
    private boolean isInPool = false;

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

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

    // Малювання кулі
    public void draw (Graphics2D g2)
    {
        g2.setColor(Color.darkGray);
        g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
    }

    // Переміщення кулі
    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;
        }

        int[][] poolsCoordinates =
((ObjectCanvas)this.canvas).getPoolsCoordinates();
```

```
        int[] poolSize = ((ObjectCanvas)this.canvas).getPoolSize();

        for (int[] poolsCoordinate : poolsCoordinates) {
            if (x + XSIZE >= poolsCoordinate[0] && x <= poolsCoordinate[0] +
poolSize[0] && y + YSIZE >= poolsCoordinate[1] && y <= poolsCoordinate[1] +
poolSize[1]) {
                this.isInPool = true;
                break;
            }
        }

        this.canvas.repaint();
    }

    // Чи була куля забита в лузу
    public boolean getIsInPool()
    {
        return this.isInPool;
    }

    // Видалення кулі
    public void delete()
    {
        ((ObjectCanvas)this.canvas).removeBall(this);
    }
}
```

Файл Pool.java

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


public class Pool
{
    private static final int XSIZE = 40;
    private static final int YSIZE = 40;

    private int x;
    private int y;

    public Pool(int x, int y, Graphics2D g2)
    {


        this.x = x - XSIZE;
        this.y = y - YSIZE;

        if (this.x < 0) this.x = 0;
        if (this.y < 0) this.y = 0;
```

```
        this.draw(g2);
    }

    public void draw (Graphics2D g2)
    {
        g2.setColor(Color.green);
        g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
    }

    public int getX()
    {
        return x;
    }

    public int getY()
    {
        return y;
    }

    public static int getXSIZE()
    {
        return XSIZE;
    }

    public static int getYSIZE()
    {
        return YSIZE;
    }
}
```

Файл ScoreListener.java

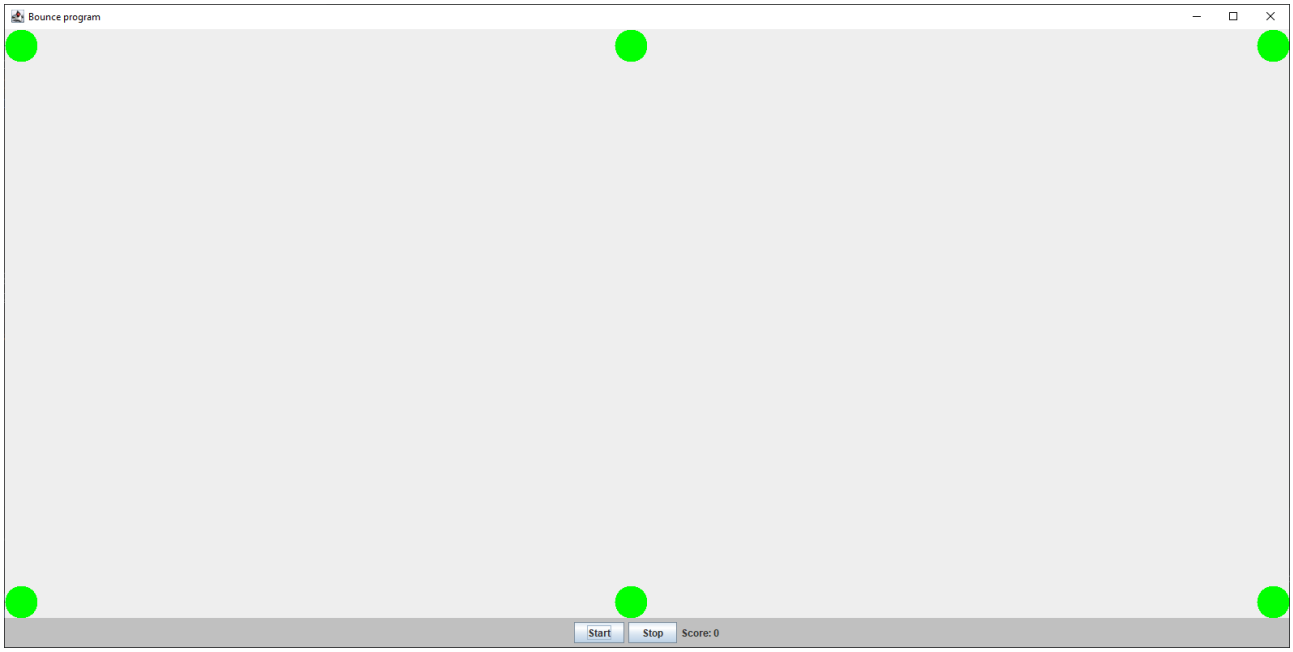
```
public interface ScoreListener
{
    void actionPerformed();
}
```

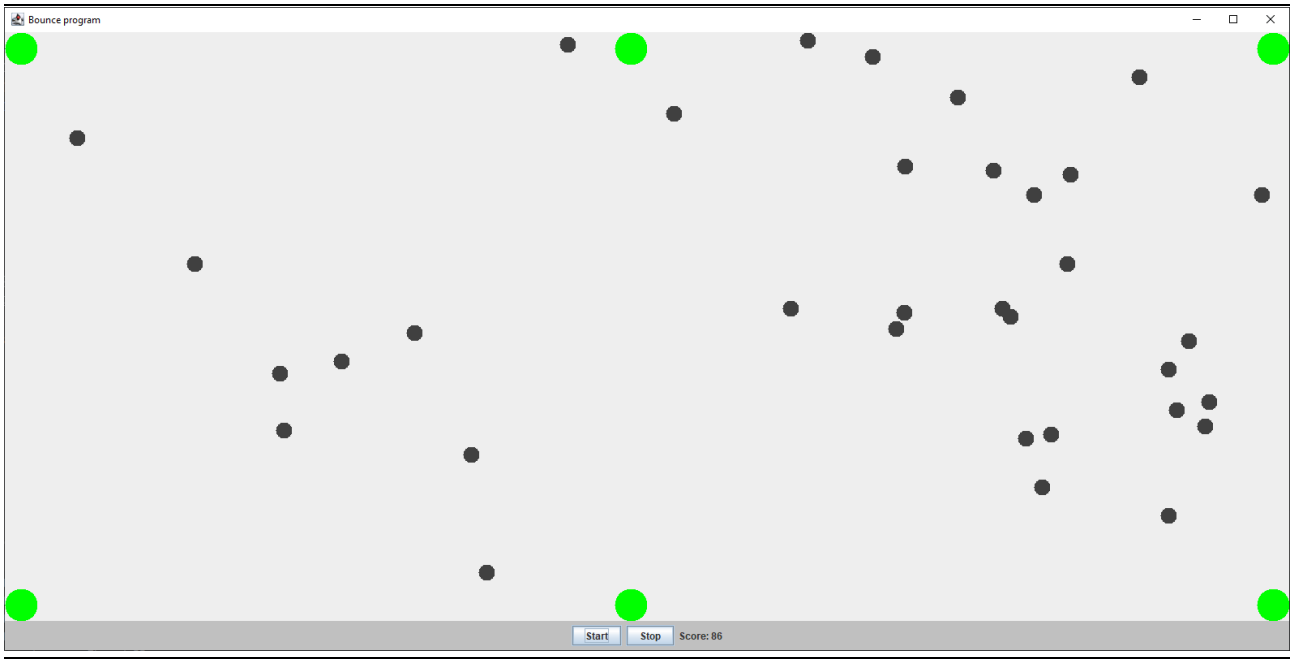
Файл Score.java

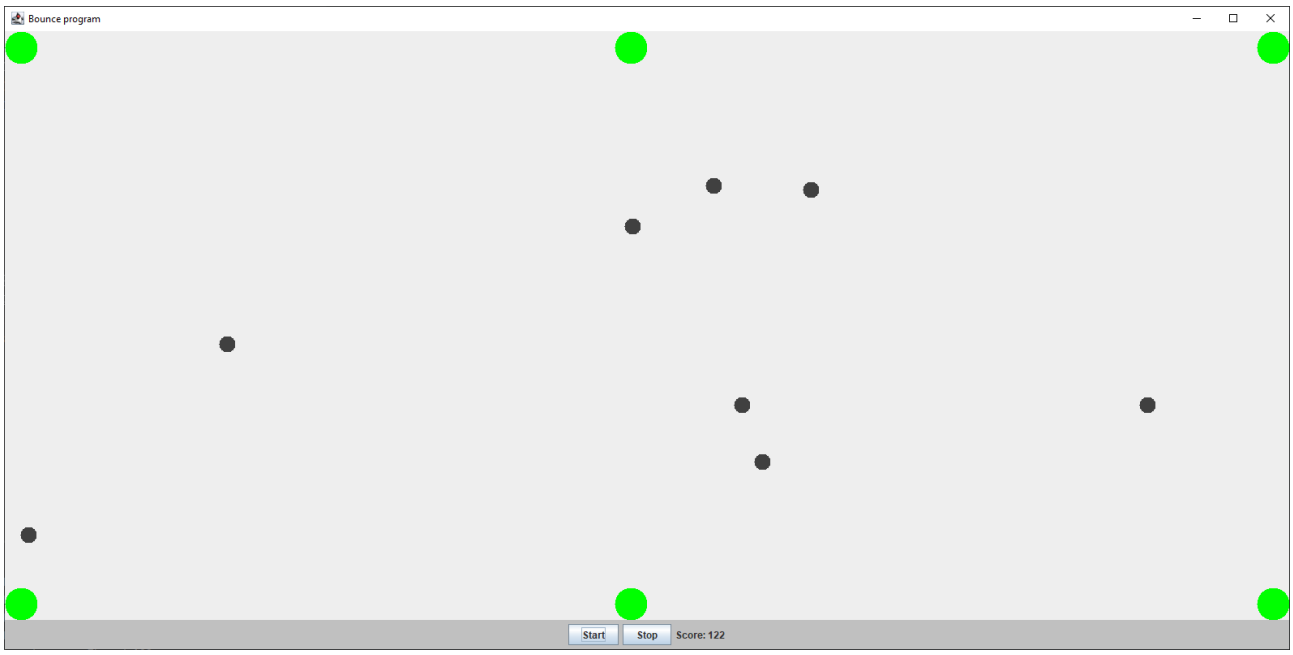
```
public class Score
{
    private static int score = 0;
    private static ScoreListener listener = null;

    public static int getScore()
    {
        return score;
    }

    public static void incrementScore()
```

```
    {
        Score.score++;
        listener.actionPerformed();
    }

    public static void addListener(ScoreListener listener)
    {
        Score.listener = listener;
    }
}
```

Скріншоти







Інші результати

```
Thread name = Thread-123
Thread name = Thread-121
Thread name = Thread-123
Thread name = Thread-121
Thread name = Thread-123
Thread name = Thread-121
Thread name = Thread-123
Thread name = Thread-121
Thread name = Thread-123
Thread name ended = Thread-123
Thread name = Thread-121
Thread name = Thread-121
Thread name = Thread-121
Thread name = Thread-121
Thread name = Thread-121
Thread name = Thread-121
Thread name = Thread-121
Thread name = Thread-121
```

Висновки

В ході виконання даного завдання було модифіковано програму зі Завдання 1, додавши лунки та реалізацію зникнення кульок при потраплянні в лунку. Також було додано динамічний каунтер, що рахує кульки, котрі потрапили до лунок. Дана модифікація забезпечила більш цікаву взаємодію з користувачем.

Завдання 3

Завдання

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.

Лістинги програми

Файл Main.java

```
import javax.swing.*;


public class Main
{
    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());
    }
}
```

Файл BounceFrame.java

```
import javax.swing.*;
import java.awt.*;


public class BounceFrame extends JFrame
{
    private final ObjectCanvas canvas;

    public static final int blueCount = 50;
    public static final int redCount = 1;

    public static final int WIDTH = 1600;
    public static final int HEIGHT = 800;
```

```
    public BounceFrame()
    {
        this.setSize(WIDTH, HEIGHT);
        this.setTitle("Bounce program");
        this.canvas = new ObjectCanvas();

//        System.out.println("In Frame Thread name = " +
Thread.currentThread().getName());

        Container content = this.getContentPane();
        content.add(this.canvas, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setBackground(Color.lightGray);

        JButton buttonStart = new JButton("Start");
        JButton buttonStop = new JButton("Stop");
        JLabel labelScoreCounter = new JLabel("Score: " + Score.getScore());

        Score.addListener(() -> {
            labelScoreCounter.setText("Score: " + Score.getScore());
            labelScoreCounter.repaint();
        });

        buttonStart.addActionListener(e -> {
            for (int i = 0; i < redCount; i++) {
                Ball b = new Ball(canvas);
                canvas.addRedBall(b);

                BallThread thread = new BallThread(b);
                thread.start();
                thread.setPriority(Thread.MAX_PRIORITY);

//                System.out.println("Thread name = " + thread.getName());
            }

            for (int i = 0; i < blueCount; i++) {
                Ball b = new Ball(canvas);
                canvas.addBlueBall(b);

                BallThread thread = new BallThread(b);
                thread.start();
                thread.setPriority(Thread.MIN_PRIORITY);

//                System.out.println("Thread name = " + thread.getName());
            }
        });

        buttonStop.addActionListener(e -> System.exit(0));

        buttonPanel.add(buttonStart);
```

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

Файл ObjectCanvas.java

```
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;


public class ObjectCanvas extends JPanel
{
    private final ArrayList<Ball> blueBalls = new ArrayList<>();
    private final ArrayList<Ball> redBalls = new ArrayList<>();
    private final Pool[] pools = new Pool[6];

    public void addBlueBall(Ball b)
    {
        this.blueBalls.add(b);
    }

    public void addRedBall(Ball b)
    {
        this.redBalls.add(b);
    }

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

        for (Ball b : blueBalls) {
            if (b != null) b.draw(g2, Color.BLUE);
//            b.draw(g2, Color.BLUE);
        }

        for (Ball b : redBalls) {
            if (b != null) b.draw(g2, Color.RED);
//            b.draw(g2, Color.RED);
        }

        pools[0] = new Pool(0, 0, g2);
        pools[1] = new Pool(super.getWidth() / 2, 0, g2);
        pools[2] = new Pool(super.getWidth(), 0, g2);
        pools[3] = new Pool(0, super.getHeight(), g2);
        pools[4] = new Pool(super.getWidth() / 2, super.getHeight(), g2);
        pools[5] = new Pool(super.getWidth(), super.getHeight(), g2);
```

```
    }

    public int[][] getPoolsCoordinates()
    {
        int[][] coordinates = new int[pools.length][2];

        for (int i = 0; i < pools.length; i++) {
            coordinates[i][0] = pools[i].getX();
            coordinates[i][1] = pools[i].getY();
        }

        return coordinates;
    }

    public int[] getPoolSize()
    {
        int[] size = new int[2];
        size[0] = Pool.getXSIZE() / 2;
        size[1] = Pool.getYSIZE() / 2;
        return size;
    }

    public synchronized void removeBall(Ball b)
    {
        Score.incrementScore();
        blueBalls.remove(b);
        redBalls.remove(b);
    }
}
```

Файл BallThread.java

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

    // Конструктор класса
    public BallThread(Ball ball)
    {
        b = ball;
    }

    @Override
    public void run()
    {
        try {
            while (!Thread.interrupted()) {
                b.move();
//                System.out.println("Thread name = " +
Thread.currentThread().getName());
                if (b.getIsInPool()) {
```

```
//                    System.out.println("Thread name ended = " +
Thread.currentThread().getName());
                    b.delete();
                    Thread.currentThread().interrupt();
                }
                Thread.sleep(5);
            }
        } catch (InterruptedException ignored) {}
    }
}
```

Файл Ball.java

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


public class Ball
{
    private final Component canvas;

    // Розміри кулі
    private static final int XSIZE = 20;
    private static final int YSIZE = 20;

    // Позиція кулі
    private int x;
    private int y;

    // Вектор кулі
    private int dx = 10;
    private int dy = 10;

    // Чи була куля забита в лузу
    private boolean isInPool = false;

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

        this.x = 400;
        this.y = 450;

//        x = new Random().nextInt(this.canvas.getWidth());
//        y = new Random().nextInt(this.canvas.getHeight());
    }

    // Малювання кулі
    public void draw (Graphics2D g2, Color color)
    {
```

```
        g2.setColor(color);
        g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
        this.canvas.repaint();
    }

    // Переміщення кулі
    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;
        }

        int[][] poolsCoordinates =
((ObjectCanvas)this.canvas).getPoolsCoordinates();
        int[] poolSize = ((ObjectCanvas)this.canvas).getPoolSize();

        for (int[] poolsCoordinate : poolsCoordinates) {
            if (x + XSIZE >= poolsCoordinate[0] && x <= poolsCoordinate[0] +
poolSize[0] && y + YSIZE >= poolsCoordinate[1] && y <= poolsCoordinate[1] +
poolSize[1]) {
                this.isInPool = true;
                break;
            }
        }

        this.canvas.repaint();
    }

    // Чи була куля забита в лузу
    public boolean getIsInPool()
    {
        return this.isInPool;
```

```
    }

    // Видалення кулі
    public void delete()
    {
        ((ObjectCanvas)this.canvas).removeBall(this);
        this.canvas.repaint();
    }
}
```

Файл Pool.java

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


public class Pool
{
    private static final int XSIZE = 40;
    private static final int YSIZE = 40;

    private int x;
    private int y;

    public Pool(int x, int y, Graphics2D g2)
    {

        this.x = x - XSIZE;
        this.y = y - YSIZE;

        if (this.x < 0) this.x = 0;
        if (this.y < 0) this.y = 0;

        this.draw(g2);
    }

    public void draw (Graphics2D g2)
    {
        g2.setColor(Color.green);
        g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
    }

    public int getX()
    {
        return x;
    }

    public int getY()
    {
        return y;
    }
```

```
    public static int getXSIZE()
    {
        return XSIZE / 2;
    }

    public static int getYSIZE()
    {
        return YSIZE / 2;
    }
}
```

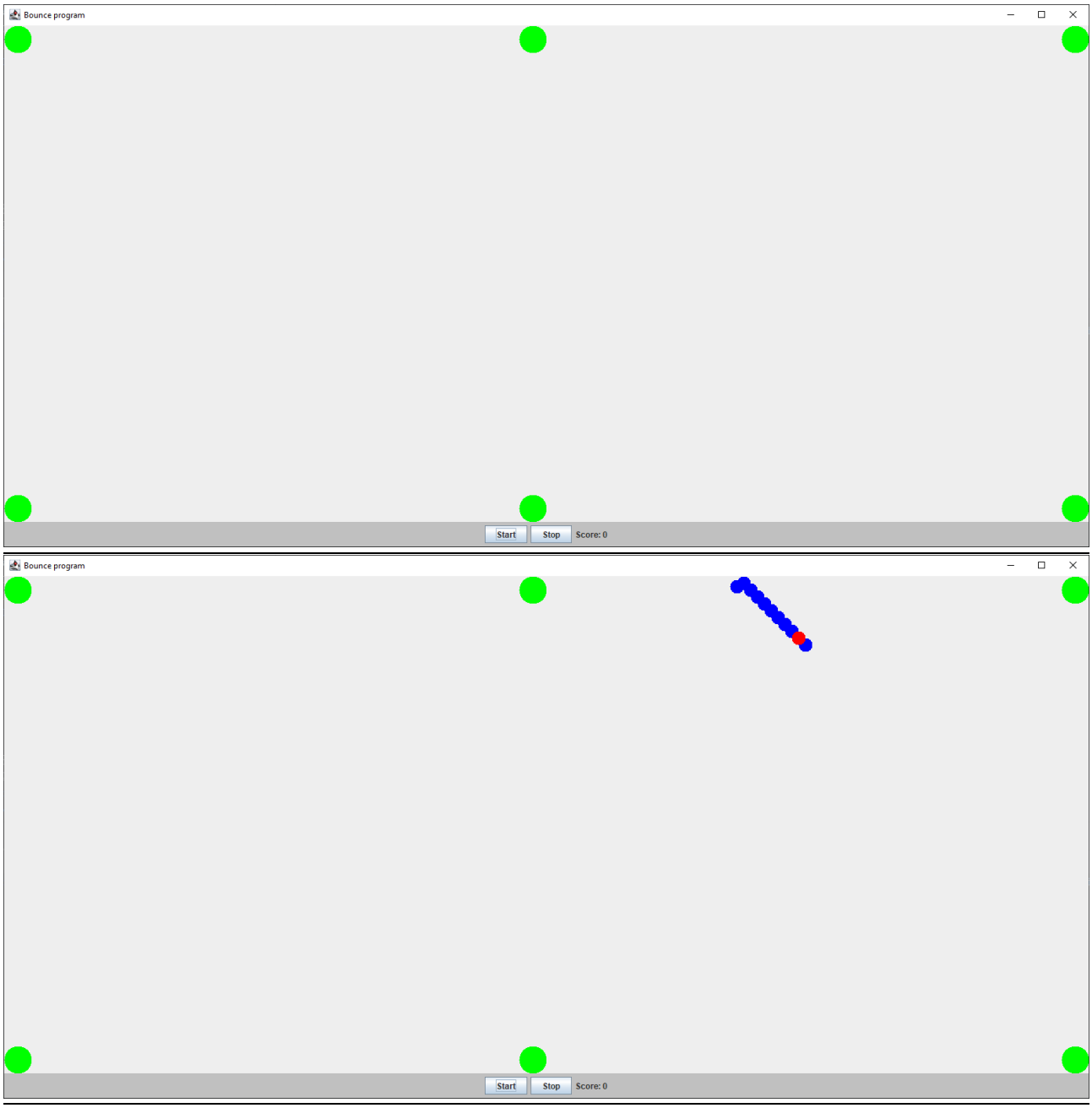
Файл ScoreListener.java

```
public interface ScoreListener
{
    void actionPerformed();
}
```

Файл Score.java

```
public class Score
{
    private static int score = 0;
    private static ScoreListener listener = null;

    public static int getScore()
    {
        return score;
    }

    public static void incrementScore()
    {
        Score.score++;
        listener.actionPerformed();
    }

    public static void addListener(ScoreListener listener)
    {
        Score.listener = listener;
    }
}
```

Скріншоти



Інші результати

Інших результатів програмою не передбачено.

Висновки

В ході виконання даного завдання було реалізовано програму, що досліджує параметри пріоритетності конкурування потоків. Було створено декілька кульок/потоків синього кольору із низьким пріоритетом та одну кульку червоного кольору із високим пріоритетом. Провівши експеримент можна зазначити, що переважно червона куля рухається швидше за сині, тобто потік червоної кулі обробляється пріоритетніше за інші потоки.

Завдання 4

Завдання

Побудуйте ілюстрацію методу join() класу Thread через взаємодію потоків, що відтворюють рух більярдних кульок різного кольору. Поясніть результат, який спостерігається

Лістинги програми

Файл Main.java

```
import javax.swing.*;


public class Main
{
    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());
    }
}
```

Файл BounceFrame.java

```
import javax.swing.*;
import java.awt.*;


public class BounceFrame extends JFrame
{
    private final ObjectCanvas canvas;

    public static final int ballsCount = 50;

    public static final int WIDTH = 1600;
    public static final int HEIGHT = 800;

    public BounceFrame()
    {
        this.setSize(WIDTH, HEIGHT);
        this.setTitle("Bounce program");
        this.canvas = new ObjectCanvas();

        System.out.println("In Frame Thread name = " +
Thread.currentThread().getName());

        Container content = this.getContentPane();
```

```
        content.add(this.canvas, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel();
        buttonPanel.setBackground(Color.lightGray);

        JButton buttonStart = new JButton("Start");
        JButton buttonStop = new JButton("Stop");

        buttonStart.addActionListener(e -> {
            Thread t = new Thread(new Runnable() {
                public void run() {
                    for (int i = 0; i < ballsCount; i++) {
                        Ball b = new Ball(canvas);
                        canvas.addBall(b);

                        BallThread thread = new BallThread(b);
                        thread.start();

                        System.out.println("Thread name = " + thread.getName());

                        try {
                            thread.join();
                        } catch (InterruptedException ignored) { }
                    }
                }
            });
            t.start();
        });

        buttonStop.addActionListener(e -> System.exit(0));

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

Файл ObjectCanvas.java

```
import javax.swing.*;
import java.awt.*;
import java.util.ArrayList;


public class ObjectCanvas extends JPanel
{
    private final ArrayList<Ball> balls = new ArrayList<>();

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

```
    }

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

        for (int i = 0; i < balls.size(); i++) {
            Ball b = balls.get(i);
            if (b != null) b.draw(g2);
//              b.draw(g2, Color.RED);
        }
    }
}
```

Файл BallThread.java

```
public class BallThread extends Thread
{
    private final Ball b;
    private final int time = 100;

    // Конструктор класса
    public BallThread(Ball ball)
    {
        b = ball;
    }

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

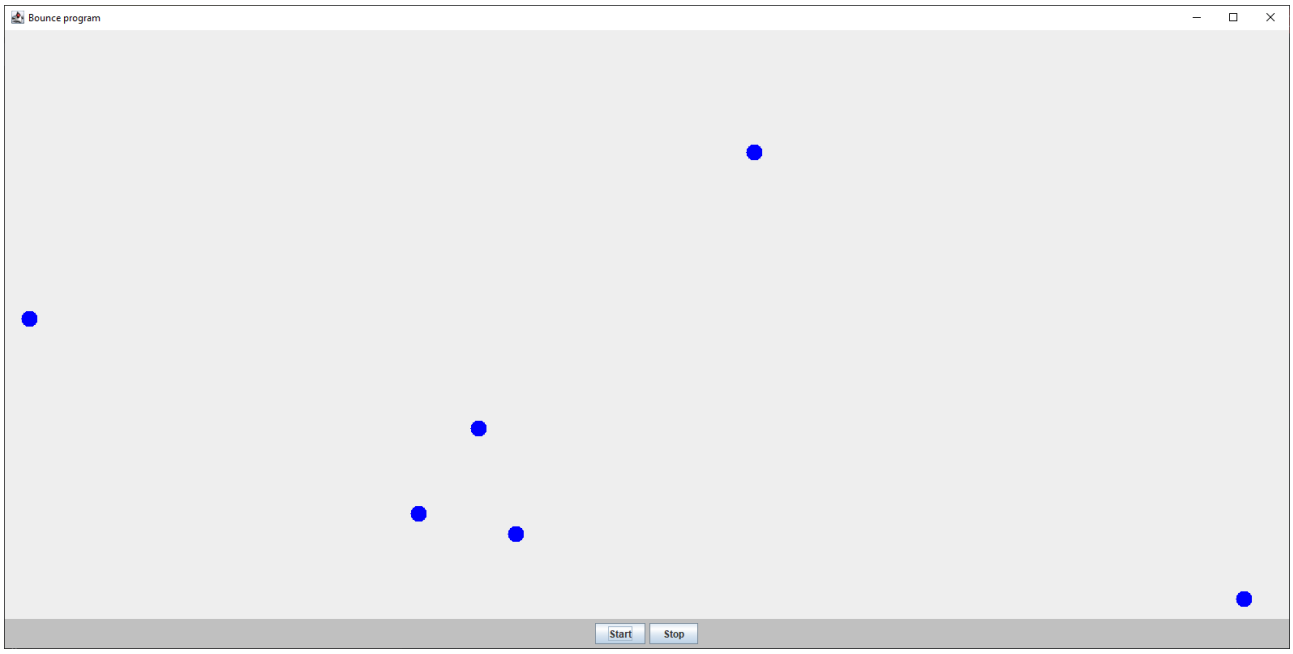
Файл Ball.java

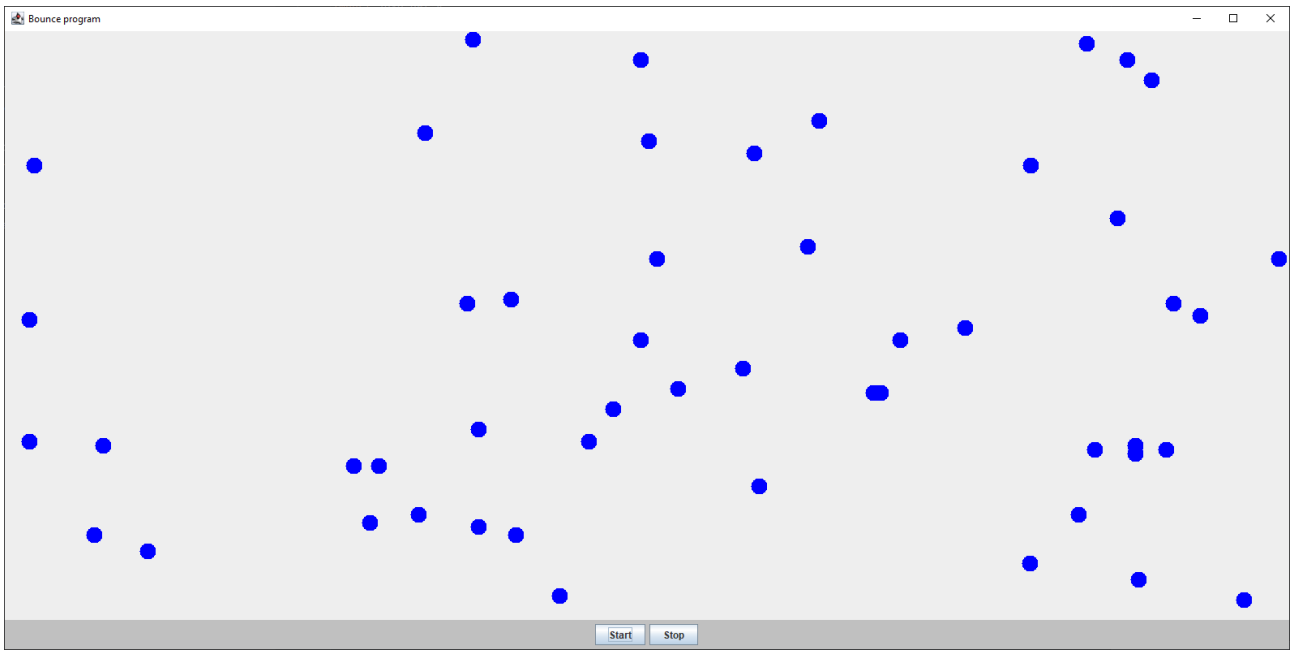
```
import java.awt.*;
import java.awt.geom.Ellipse2D;
import java.util.Random;


public class Ball
```

```
{
    private final Component canvas;

    // Розміри кулі
    private static final int XSIZE = 20;
    private static final int YSIZE = 20;

    // Позиція кулі
    private int x;
    private int y;

    // Вектор кулі
    private int dx;
    private int dy;

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

        x = new Random().nextInt(this.canvas.getWidth());
        y = new Random().nextInt(this.canvas.getHeight());

        if (Math.random() < 0.5) {
            if (Math.random() < 0.5) {
                dx = 10;
                dy = 10;
            } else {
                dx = 10;
                dy = -10;
            }
        } else {
            if (Math.random() < 0.5) {
                dx = -10;
                dy = 10;
            } else {
                dx = -10;
                dy = -10;
            }
        }
    }

    // Малювання кулі
    public void draw (Graphics2D g2)
    {
        g2.setColor(Color.BLUE);
        g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
        this.canvas.repaint();
    }

    // Переміщення кулі
```

```
    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();
    }
}
```

Скріншоти





Інші результати

Інших результатів виконання програми не передбачено

Висновки

В ході виконання даного завдання було реалізовано ілюстрацію методу join() класу Thread через взаємодію потоків, що відтворюють рух більярдних кульок. Програма оброблює 100 раз один потік та переходить до виконання наступного. Наступний потік очікує допоки попередній не завершить роботу.

Завдання 5

Завдання

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. Використовуючи найпростіші методи управління потоками, добийтесь почергового виведення на консоль символів.

Лістинги програми

Файл Main.java

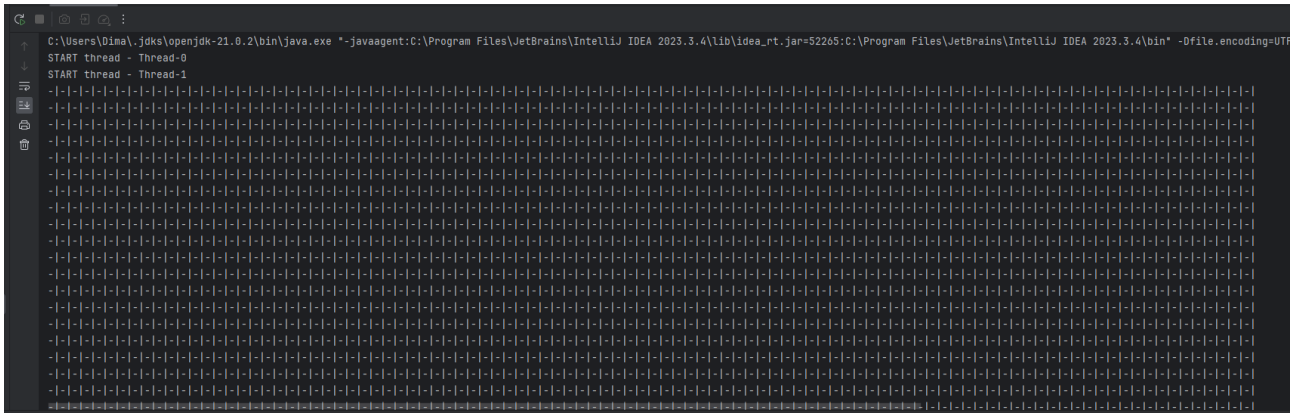
```
public class Main
{
    public static void main(String[] args) throws InterruptedException
    {
        MyThread thread1 = new MyThread("-");
        thread1.start();
        thread1.join();

        MyThread thread2 = new MyThread("|");
        thread2.start();
        thread2.join();

        for (int i = 0; i < 100; i++) {
            for (int j = 0; j < 100; j++) {
                thread1.print();
                thread1.join();

                thread2.print();
                thread2.join();
            }
            System.out.println();
        }
    }
}
```

Файл MyThread.java

```
public class MyThread extends Thread
{
    private final String output_str;

    public MyThread (String s)
    {
        this.output_str = s;
    }
```

```
    @Override
    public void run()
    {
        System.out.println("START thread - " + Thread.currentThread().getName());
    }

    public void print()
    {
        try {
            System.out.print(this.output_str);
            Thread.sleep(0);
        } catch (InterruptedException ignored) { }
    }
}
```

Скріншоти



Інші результати

Інших результатів ходом виконання завдання не передбачено

Висновки

В ході виконання даного завдання було створено програму, що по черзі виводить символи в різних потоках. При простому створенні потоків, потоки конкурують і ми маємо випадковий набір символів «|» і «-». Це відбувається через те, що потоки працюють паралельно і не синхронізовані між собою. Проте за допомогою join ми модифікували програму так, аби один потік очікував другий та мати такий результат, аби по черзі символи виводились.

Завдання 6

Завдання

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. Використовуючи синхронізований доступ, добийтесь правильної роботи лічильника при одночасній роботі з ним двох і більше потоків. Опрацюйте використання таких способів синхронізації: синхронізований метод, синхронізований блок, блокування об’єкта. Порівняйте способи синхронізації.

Лістинги програми

Файл Main.java

```
public class Main
{
    public static void main(String[] args) throws InterruptedException
    {
        System.out.println("No sync ");
        Thread threadInc = new Thread(() -> {
            for (int i = 0; i < 100000; i++) {
                Counter.increment();
            }
        });

        Thread threadDec = new Thread(() -> {
            for (int i = 0; i < 100000; i++) {
                Counter.decrement();
            }
        });

        threadInc.start();
        threadDec.start();

        threadInc.join();
        threadDec.join();

        System.out.println("Counter: " + Counter.getI());

        Counter.reset();

        System.out.println("Sync by method");

        Thread threadIncSync = new Thread(() -> {
            for (int i = 0; i < 100000; i++) {
                Counter.incrementSync();
```

```
        }
    });

    Thread threadDecSync = new Thread(() -> {
        for (int i = 0; i < 100000; i++) {
            Counter.decrementSync();
        }
    });

    threadIncSync.start();
    threadDecSync.start();

    threadIncSync.join();
    threadDecSync.join();

    System.out.println("Counter: " + Counter.getI());

    Counter.reset();

    System.out.println("Sync by block");

    Thread threadIncSyncBlock = new Thread(() -> {
        for (int i = 0; i < 100000; i++) {
            Counter.incrementSyncBlock();
        }
    });

    Thread threadDecSyncBlock = new Thread(() -> {
        for (int i = 0; i < 100000; i++) {
            Counter.decrementSyncBlock();
        }
    });

    threadIncSyncBlock.start();
    threadDecSyncBlock.start();

    threadIncSyncBlock.join();
    threadDecSyncBlock.join();

    System.out.println("Counter: " + Counter.getI());

    Counter.reset();

    System.out.println("Sync by lock");

    Thread threadIncLock = new Thread(() -> {
        for (int i = 0; i < 100000; i++) {
            Counter.incrementLock();
        }
    });
```

```
        Thread threadDecLock = new Thread(() -> {
            for (int i = 0; i < 100000; i++) {
                Counter.decrementLock();
            }
        });

        threadIncLock.start();
        threadDecLock.start();

        threadIncLock.join();
        threadDecLock.join();

        System.out.println("Counter: " + Counter.getI());

        Counter.reset();
    }
}
```

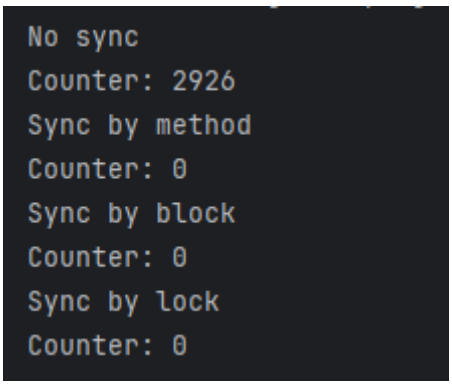
Файл Counter.java

```
public class Counter
{
    private final static Object lock = new Object();
    private static int i = 0;

    public static int getI()
    {
        return Counter.i;
    }

    public static void increment()
    {
        Counter.i++;
    }

    public static void decrement()
    {
        Counter.i--;
    }

    public static synchronized void incrementSync()
    {
        Counter.i++;
    }

    public static synchronized void decrementSync()
    {
        Counter.i--;
    }

    public static void incrementSyncBlock()
```

```
    {
        synchronized (Counter.class) {
            Counter.i++;
        }
    }

    public static void decrementSyncBlock()
    {
        synchronized (Counter.class) {
            Counter.i--;
        }
    }

    public static void incrementLock()
    {
        synchronized (lock) {
            Counter.i++;
        }
    }

    public static void decrementLock()
    {
        synchronized (lock) {
            Counter.i--;
        }
    }

    public static void reset()
    {
        Counter.i = 0;
    }
}
```

Скріншоти

```
No sync
Counter: 16181
Sync by method
Counter: 0
Sync by block
Counter: 0
Sync by lock
Counter: 0
```



Інші результати

Інші результати ходом виконання програми не передбачено

Висновки

В ході виконання даного завдання було створено програму, що реалузовує клас «Counter» з методами increment() та decrement(). Ми отримали в результаті випадкове значення, очскільки потоки між собою не синхронізовані та ми маємо «баг» пам’яті, де один поток редагує шмат пам’яті, який вже був заредагований іншим потоком. Ми синхронізували потоки трьома різними способами. Найбільш зручним мені здався спосіб з синхронізацією методів та спосіб з синхронізацією блоком.