МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»

Факультет управления и информатики в технологических системах
Кафедра Информационной безопасности
Направление подготовки
(специальность) 10.05.03 Информационная безопасность автоматизированных систем

Отчет

по практике по технологиям и методам программирования
наименование (вид) практики

Выполнил студент гр. УБ-42
Крылов Никита Романович
(Ф.И.О.)

_____
_____
(подпись)

Проверили:

Маслов А.А.
(Ф.И.О.)

_____ _____
(оценка) *(подпись)*

_____
(дата)

Воронеж - 2025

Задание 15.

Модифицировать код программы bouncingcircle таким образом, чтобы вместо круга движение осуществлял экземпляр реализованного ранее (в лабораторной работе No2) класса ColorableRect(), как показано ниже: (BouncingBox). о Модифицировать код предыдущей программы таким образом, чтобы движение осуществляли сразу 10 экземпляров класса Rectangle, 10 класса DrawableRect и 10 экземпляров класса ColorableRect. Все созданные объекты должны хранится в одном массиве с типом класса родителя Rectangle. Ниже показано, как должен выглядеть апплет. (BouncingBox2).

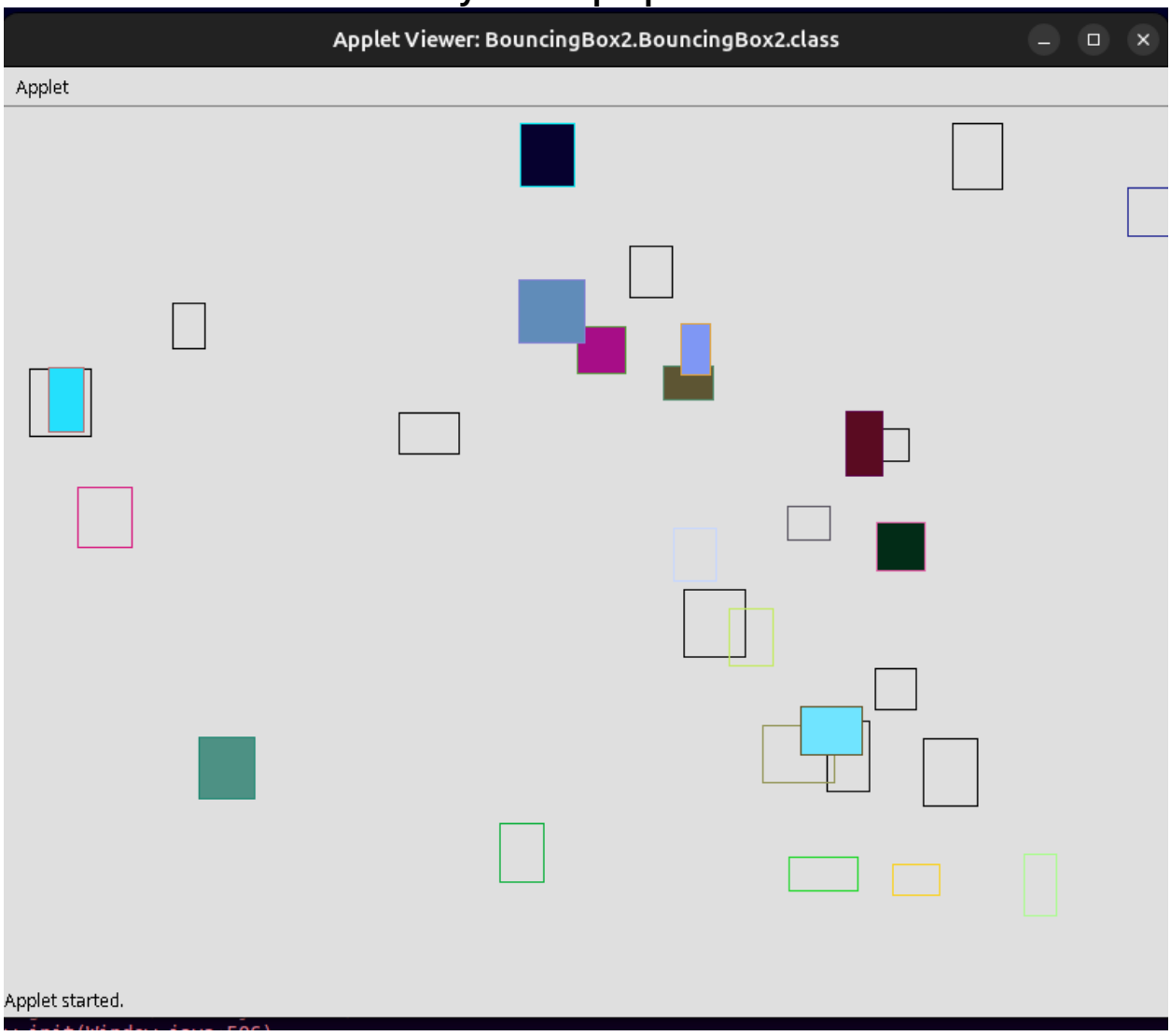
Код программы:

```
package BouncingBox2;

import java.applet.*;
import java.awt.*;
import java.util.Random;

public class BouncingBox2 extends Applet implements Runnable {
private Rectangle[] shapes = new Rectangle[30];
private int[] dx = new int[30];
private int[] dy = new int[30];
private Thread animator;
private volatile boolean pleaseStop;
private Random random = new Random();

public void init() {
for (int i = 0; i < 10; i++) {
// 10 Rectangle
shapes[i] = new Rectangle(
random.nextInt(200), random.nextInt(200),
20 + random.nextInt(30), 20 + random.nextInt(30));
dx[i] = 1 + random.nextInt(5);
dy[i] = 1 + random.nextInt(5);

// 10 DrawableRect
shapes[i+10] = new DrawableRect(
random.nextInt(200), random.nextInt(200),
20 + random.nextInt(30), 20 + random.nextInt(30),
new Color(random.nextInt(256), random.nextInt(256), random.nextInt(256)));

// 10 ColorableRect
shapes[i+20] = new ColorableRect(
random.nextInt(200), random.nextInt(200),
20 + random.nextInt(30), 20 + random.nextInt(30),
new Color(random.nextInt(256), random.nextInt(256), random.nextInt(256)),
new Color(random.nextInt(256), random.nextInt(256), random.nextInt(256)));

dx[i+10] = 1 + random.nextInt(5);
dy[i+10] = 1 + random.nextInt(5);
dx[i+20] = 1 + random.nextInt(5);
dy[i+20] = 1 + random.nextInt(5);
}
}

public void paint(Graphics g) {
for (Rectangle shape : shapes) {
if (shape instanceof ColorableRect) {
((ColorableRect)shape).draw(g);
} else if (shape instanceof DrawableRect) {
```

```
((DrawableRect)shape).draw(g);
} else {
g.setColor(Color.black);
g.drawRect(shape.x, shape.y, shape.width, shape.height);
}
}
}

public void animate() {
Rectangle bounds = getBounds();
for (int i = 0; i < shapes.length; i++) {
if ((shapes[i].x + dx[i] < 0) ||
(shapes[i].x + shapes[i].width + dx[i] > bounds.width)) dx[i] = -dx[i];
if ((shapes[i].y + dy[i] < 0) ||
(shapes[i].y + shapes[i].height + dy[i] > bounds.height)) dy[i] = -dy[i];
shapes[i].x += dx[i];
shapes[i].y += dy[i];
}
repaint();
}

public void run() {
while(!pleaseStop) {
animate();
try { Thread.sleep(50); }
catch(InterruptedException e) {}
}
}

public void start() {
animator = new Thread(this);
pleaseStop = false;
animator.start();
}

public void stop() {
pleaseStop = true;
}
}

package BouncingBox2;

import java.awt.*;

public class DrawableRect extends java.awt.Rectangle {
private Color color;

public DrawableRect(int x, int y, int w, int h, Color c) {
super(x, y, w, h);
this.color = c;
}

public void draw(Graphics g) {
g.setColor(color);
g.drawRect(x, y, width, height);
}
}



package BouncingBox2;

import java.awt.*;
```

```
public class ColorableRect extends java.awt.Rectangle {
private Color borderColor;
private Color fillColor;

public ColorableRect(int x, int y, int w, int h, Color b, Color f) {
super(x, y, w, h);
this.borderColor = b;
this.fillColor = f;
}

public void draw(Graphics g) {
g.setColor(fillColor);
g.fillRect(x, y, width, height);
g.setColor(borderColor);
g.drawRect(x, y, width, height);
}
}
```

Результат программы.



Вывод

Вывод отчета

В ходе выполнения работы я изучил основы создания анимации в Java с использованием апплетов, работу с графическими объектами и многопоточностью для организации плавного движения. Программа BouncingBox2 наглядно демонстрирует принципы обработки отрисовки, управления объектами и взаимодействия потоков.

1. Работа с графикой и объектами
Программа использует классы Rectangle,DrawableRect и ColorableRect для отображения различных фигур, что иллюстрирует следующие аспекты:
- Наследование и полиморфизм– разные типы прямоугольников обрабатываются единообразно благодаря проверке типов (`instanceof`) и переопределению метода `draw()`.
- Динамическое поведение– объекты перемещаются с разными скоростями, заданными случайным образом, что делает анимацию более разнообразной.
- Гибкость отрисовки– в зависимости от типа фигуры применяется разный стиль: обычный прямоугольник, закрашенный или с границей и заливкой разных цветов.

2.Организация анимации и многопоточности
- Цикл анимации – метод `animate()` обновляет координаты объектов, проверяет столкновения с границами окна и вызывает перерисовку.
- Работа в отдельном потоке– класс реализует `Runnable`, что позволяет анимации выполняться без блокировки интерфейса. Использование флага `pleaseStop` обеспечивает корректное завершение потока.
-Контроль скорости– задержка `Thread.sleep(50)` регулирует частоту обновления кадров, делая движение плавным.

3. Практическая значимость
Программа охватывает несколько ключевых аспектов Java:
- Обработка графики– работа с `Graphics`, отрисовка фигур и управление их свойствами.
- Многопоточность– применение `Thread` для фонового выполнения анимации.
ООП-принципы– использование наследования, полиморфизма и инкапсуляции для структурирования кода.

Таким образом, проект успешно демонстрирует создание интерактивной анимации в Java, сочетая работу с графикой, многопоточностью и объектно-ориентированным программированием. Полученные знания могут быть применены для разработки более сложных визуальных приложений, игр и симуляторов.