Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчёт**

**по лабораторной работе №4**

**Дисциплина: Разработка мобильных приложений**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. А. Курбатский

Направление подготовки 02.03.02 Фундаментальная информатика и

информационные технологии

Направленность (профиль) Математическое и программное обеспечение

компьютерных технологий

Преподаватель

д-р техн. наук, проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. И. Шиян

**Цель работы:** освоить методы разработки с графическим интерфейсом Java, ознакомиться с различными компоновщиками. Научиться обрабатывать события пользовательского интерфейса.

**Общие сведения о графических интерфейсах**

Графический пользовательский интерфейс (GUI) – основной способ взаимодействия конечных пользователей с java-приложением. Для разработки прикладного программного обеспечения на языке Java, а точнее графического интерфейса приложений, обычно используются пакеты AWT и Swing. AWT – Abstract Window Toolkit (для доступа загружается пакет java.awt) – содержит набор классов, позволяющих выполнять графические операции и создавать оконные элементы управления, подобно тому, как это делается в VBA и Delphi. Swing (для доступа загружается пакет javax.swing) является «надстройкой» над AWT и содержит новые классы, в основном аналогичные AWT. К именам классов добавляется J (JButton, JLabel и др.).

Возвращаясь к Swing, отметим, что на данный момент основные классы для построения визуальных интерфейсов содержатся в пакете javax.swing. Из пакета AWT используются классы для обработки сообщений. Никаких препятствий для совместного использования графических компонентов этих двух библиотек нет.

**Способы размещения элементов**

Размещение элементов с помощью менеджеров (компоновщиков) Можно, конечно, размещать компоненты «вручную», задавая их размеры и положение в контейнере абсолютными координатами в координатной системе контейнера и используя метод setBounds(). Такой способ размещает компоненты с точностью до пикселя, но не позволяет перемещать их менеджеру компонентов автоматически при изменении размеров окна с помощью мыши компоненты останутся на своих местах привязанными к левому верхнему углу контейнера. Кроме того, нет гарантии, что все мониторы отобразят компоненты так, как вы задумали. Чтобы учесть изменение размеров окна, надо задать размеры и положение компонента относительно размеров контейнера, но тогда при всяком изменении размеров окна расположение компонента придется задавать заново. Чтобы избавить программиста от этой кропотливой работы, в библиотеку AWT внесены два интерфейса: LayoutManager и порожденный от него интерфейс LayoutManager2, а также несколько реализаций этих интерфейсов: классы BorderLayout, CardLayout, FlowLayout, GridLayout, GridBagLayou и др. Эти классы названы менеджерами размещения компонентов (layout manager). Библиотека Swing добавляет к указанным классам свои менеджеры размещения, используемые контейнерами Swing.

**События и их обработка**

Рассмотрим общую схему обработки событий в Java. Она основана на архитектурном шаблоне «Наблюдатель» и носит название «Event-Driven» (или «Event-Delegation») – событийная модель и используется для обработки событий в большинстве языков визуального программирования. Пользователь воздействут на исходный объект (например, кнопка – Button или поле ввода – Textfield). При срабатывании исходный объект создает объект события для захвата действия (например, щелчок мышью по x и y, введенные тексты и т. п.). Этот объект события будет передан всем зарегистрированным объектам-слушателям, и соответствующий метод обработчика события будет вызван для выполнения необходимого действия. Чтобы «выразить интерес» к событию определенного источника, слушатели должны быть зарегистрированы в источнике. Другими словами, слушатель (и) «подписывается» на событие источника, а источник «публикует» событие всем своим подписчикам при активации. Такая схема еще известна как шаблон «подписки-публикации».

**Ход работы:**

В ходе работы было создано приложение для игры в игру «Balls».

Интерфейс игры представляет собой объект JFrame на котором с помощью BoxLayout размещено два элемента: JLabel для отображения текущего счета, JPanel выступающий в виде игрового поля. Поле игры представлено в виде элемента JPanel, который содержит 81 кнопку. Компоновщиком выступает GridLayout с 9 строками и 9 столбцами.

В начале игры в случайным образом загораются 3 кружка, цвет выбирается также случайно.

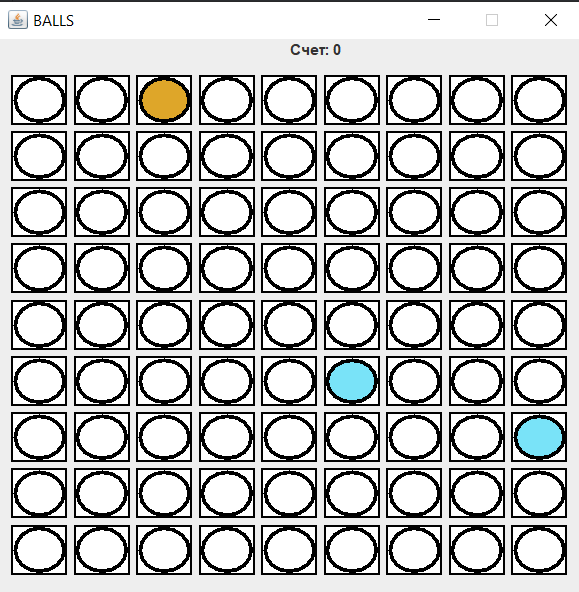


Рисунок 1 - Вид приложения при запуске

Класс игрового поля Battlefield реализует интерфейс ActionListener, поэтому он выступает обработчиком событий, для кнопок, которые содержит.

Каждый ход игрок выбирает кружок и переставляет его на другое место путем нажатия на соответствующие ячейки. Игра заканчивается когда поле будет полностью заполнено об этом информирует MessageBox с соответствующим сообщением.

**Вывод:** были освоены методы разработки с графическим интерфейсом Java. Научился обрабатывать события пользовательского интерфейса.

**Листинг программы:**

**Файл Battlefield.java**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.\*;  
import java.util.List;  
import java.util.stream.IntStream;  
  
public class Battlefield extends JPanel implements ActionListener  
{  
 Component[] comp;  
 Random rand = new Random();  
  
 List<Integer> whiteButtons;  
  
 private int \_demention;  
  
 int score = 0;  
  
 Color[] colorList = new Color[] {  
 new Color(121, 227, 248),  
 new Color(161, 166, 250),  
 new Color(229, 83, 124),  
 new Color(222, 166, 41),  
 new Color(31, 248, 155)  
 };  
  
 CircleButton prevButton = null;  
 JLabel l = null;  
 public Battlefield(int n, JLabel label)  
 {  
 super();  
 l = label;  
 \_demention = n;  
 setBorder(BorderFactory.*createEmptyBorder*(10,10,10,10));  
 setLayout(new GridLayout(9,9,5,5));  
 whiteButtons = new ArrayList<>(IntStream.*rangeClosed*(0, 80).boxed().toList());  
 for(int i = 0; i < n \* n; i++)  
 {  
 JButton b = new CircleButton(Color.*WHITE*, i);  
 b.addActionListener(this);  
 add(b);  
 }  
 comp = getComponents();  
 randomThree();  
 }  
  
 public void randomThree()  
 {  
 Collections.*shuffle*(whiteButtons);  
 for(int i = 0; i < 3; i++)  
 {  
 ((CircleButton)comp[whiteButtons.get(i)]).setColor(colorList[rand.nextInt(0,5)]);  
 whiteButtons.remove(i);  
 }  
 }  
  
 public void searchFifty(int num)  
 {  
 int i = num / \_demention;  
 int j = num % \_demention;  
  
 int count = 1;  
 List<Integer> list = new ArrayList<>();  
  
 CircleButton prevB = ((CircleButton) comp[j]);  
 Color tempColor = prevB.getColor();  
 list.add(prevB.getNumber());  
  
 for(int k = j + \_demention; k < \_demention \* \_demention - (\_demention - j) ; k+= \_demention)  
 {  
 CircleButton currB = ((CircleButton) comp[k]);  
 Color currColor = currB.getColor();  
 if(currColor == tempColor)  
 {  
 count++;  
 list.add(currB.getNumber());  
 }  
 else  
 {  
 count = 1;  
 tempColor = currColor;  
 list.clear();  
 list.add(currB.getNumber());  
 }  
 if(count == 5)  
 {  
 score +=5;  
 bleaching(list);  
 count = 1;  
 list.clear();  
 list.add(currB.getNumber());  
 }  
 }  
  
 count = 1;  
 list.clear();  
 prevB = ((CircleButton) comp[i]);  
 tempColor = prevB.getColor();  
 list.add(prevB.getNumber());  
  
 for(int k = 0; k < \_demention; k++)  
 {  
 CircleButton currB = ((CircleButton) comp[k + (i \* \_demention)]);  
 Color currColor = currB.getColor();  
 if(currColor == tempColor)  
 {  
 count++;  
 list.add(currB.getNumber());  
 }  
 else  
 {  
 count = 1;  
 tempColor = currColor;  
 list.clear();  
 list.add(currB.getNumber());  
 }  
 if(count == 5)  
 {  
 score +=5;  
 bleaching(list);  
 count = 1;  
 list.clear();  
 list.add(currB.getNumber());  
 }  
 }  
 l.setText(String.*format*("Cчет: %d", score));  
 }  
  
 public void bleaching(List<Integer> list)  
 {  
 for(int i = 0; i < list.size(); i++)  
 {  
 whiteButtons.add(list.get(i));  
 ((CircleButton)comp[list.get(i)]).setColor(Color.*WHITE*);  
 }  
  
 if(whiteButtons.size() == 0){  
 JOptionPane.*showMessageDialog*(this, String.*format*("Ваш счет %d. Неплохо", score));  
 }  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e)  
 {  
 if(prevButton != null)  
 {  
 CircleButton ourButton = ((CircleButton)e.getSource());  
 if(prevButton.getColor() == Color.*WHITE* && ourButton.getColor() != Color.*WHITE*)  
 {  
 whiteButtons.remove((Integer) prevButton.getNumber());  
 whiteButtons.add(ourButton.getNumber());  
 prevButton.setColor(ourButton.getColor());  
 ourButton.setColor(Color.*WHITE*);  
 randomThree();  
 searchFifty(prevButton.getNumber());  
 prevButton = null;  
 }  
 else if(ourButton.getColor() == Color.*WHITE* && prevButton.getColor() != Color.*WHITE*)  
 {  
 whiteButtons.remove((Integer) ourButton.getNumber());  
 whiteButtons.add(prevButton.getNumber());  
 ourButton.setColor(prevButton.getColor());  
 prevButton.setColor(Color.*WHITE*);  
 prevButton = null;  
 randomThree();  
 searchFifty(ourButton.getNumber());  
 }  
 else  
 {  
 prevButton = null;  
 }  
 }  
 else  
 {  
 prevButton = (CircleButton) e.getSource();  
 }  
 }  
  
  
  
}

**Файл CircleButton.java**

import javax.swing.\*;  
import java.awt.\*;  
  
  
public class CircleButton extends JButton  
{  
 private Color \_c;  
 private int \_number;  
  
 public Color getColor()  
 {  
 return \_c;  
 }  
  
 public int getNumber(){ return \_number; }  
  
 public void setColor(Color c)  
 {  
 \_c = c;  
 repaint();  
 }  
  
 public CircleButton(Color color, int num)  
 {  
 super();  
 \_c = color;  
 \_number = num;  
 setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*, 2));  
 setBackground(Color.*WHITE*);  
  
 }  
 @Override  
 protected void paintComponent(Graphics g)  
 {  
 super.paintComponent(g);  
 Graphics2D g2d = (Graphics2D) g;  
 g2d.setColor(\_c);  
 g2d.fillOval(3, 3, getWidth()-6, getHeight()-6);  
 g2d.setColor(Color.*BLACK*);  
 g2d.setStroke(new BasicStroke(3));  
 g2d.drawOval(3, 3 , getWidth() - 6, getHeight() - 6);  
 }  
  
}

**Файл Main.java**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.util.Random;  
  
  
  
public class Main {  
 public static void main(String[] args) {  
 JFrame f = new JFrame("BALLS");  
 f.setSize(480, 480);  
 f.setResizable(false);  
 f.setLayout(new BoxLayout(f.getContentPane(), BoxLayout.*Y\_AXIS*));  
 JLabel score = new JLabel("Счет: 0");  
 score.setHorizontalTextPosition(JLabel.*CENTER*);  
 JPanel p = new Battlefield(9, score);  
 f.add(score);  
 f.add(p);  
 f.setLocationRelativeTo(null);  
 f.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 f.setVisible(true);  
 }  
  
  
}