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**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Дисциплина: Программирование для мобильных платформ**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.Д. Воробьев

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Т.А. Приходько

**Тема работы:** Разработка графического приложения игры лото с использование многопоточности.

**Постановка задачи:**

Четыре игрока строго по очереди выставляют свои фишки (у каждого свой цвет фишки) на общую доску, о чем каждый выдает сообщение, например: «Красный, фишка З». По окончании (когда все клетки заполнены) сообщается, кто закончил первым.

**Описание решения:**

1. Класс LotoGame: Этот класс представляет главное окно игры. Он создает игровую доску, создает четырех игроков, инициализирует таймер и управляет ходами игроков.
2. Класс Player: Этот класс представляет игрока. У каждого игрока есть имя, цвет фишек, игровая доска и количество размещенных фишек. Игроки ходят поочередно, пытаясь разместить фишку на доске.
3. Класс Board: Этот класс представляет игровую доску. Он содержит ячейки (плитки) и отслеживает состояние игры. Для предотвращения коллизий при размещении фишек на доске, используется многопоточность и синхронизированные методы.
4. Класс LotoTile: Этот класс представляет фишку с номером и цветом.
5. В игре используется таймер (javax.swing.Timer), чтобы позволить игрокам ходить по очереди. Каждый тик таймера вызывает метод placeTile() у текущего игрока, который пытается разместить фишку на доске. После каждого хода обновляется отображение доски.
6. Игра продолжается, пока все ячейки на доске не будут заполнены. По окончании игры таймер останавливается, и объявляется победитель, тот игрок, который первым заполнил все ячейки.

На рисунке 1 представлена UML-диаграмма классов.

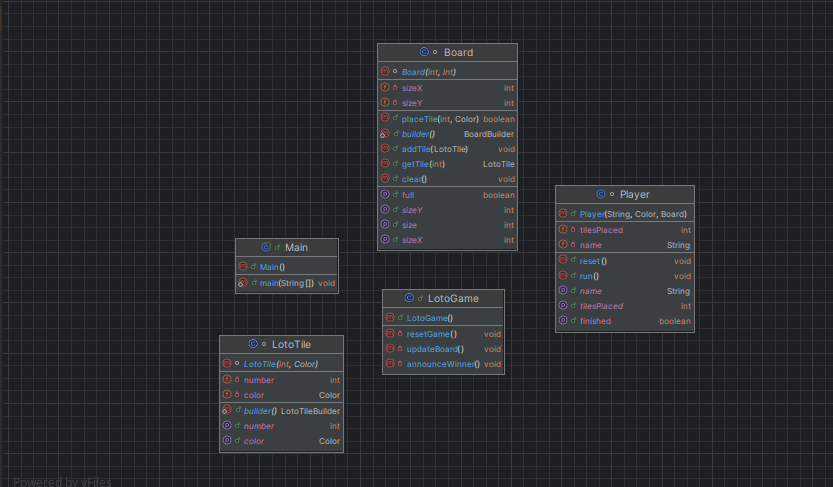


Рисунок 1 – UML-диаграмма

На рисунке 2 – 3 представлены итоги выполнения программы.



Рисунок 2 – Пример 1.

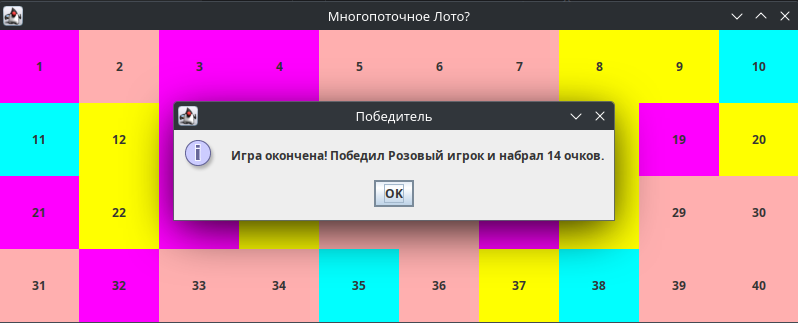


Рисунок 3 – Пример 2.

**Листинг программ**

Файл LotoGame.java

public class LotoGame extends JFrame {  
 private final Board board;  
 private final List<Player> players;  
 private final int boardSizeX = 4;  
 private final int boardSizeY = 10;  
 private final Timer gameTimer;  
  
 private final Timer restartTimer;  
 private boolean restartPending = false;  
  
 public LotoGame() {  
 board = Board.*builder*()  
 .sizeY(boardSizeY)  
 .sizeX(boardSizeX).build();  
 players = new ArrayList<>();  
 players.add(new Player("Розовый", Color.*PINK*, board));  
 players.add(new Player("Циан", Color.*CYAN*, board));  
 players.add(new Player("Магнета", Color.*MAGENTA*, board));  
 players.add(new Player("Желтый", Color.*YELLOW*, board));  
  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setSize(800, 320);  
 setLocationRelativeTo(null);  
 setLayout(new GridLayout(boardSizeX, boardSizeY));  
 setTitle("Многопоточное Лото?");  
  
 for (int i = 0; i < boardSizeY \* boardSizeX; i++) {  
 JButton button = new JButton();  
 button.setPreferredSize(new Dimension(50, 50));  
 add(button);  
 board.addTile(  
 LotoTile.*builder*()  
 .number(i + 1)  
 .build()  
 );  
  
 button.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 if (e.getClickCount() == 2) {  
 if (!restartPending) {  
 restartPending = true;  
 restartTimer.start();  
 }  
 }  
 }  
 });  
 }  
  
 gameTimer = new Timer(100, new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Random random = new Random();  
 int randomIndex;  
 do {  
 randomIndex = random.nextInt(players.size());  
 } while (players.get(randomIndex).isFinished());  
  
 Player currentPlayer = players.get(randomIndex);  
  
 SwingWorker<Void, Void> worker = new SwingWorker<>() {  
 @Override  
 protected Void doInBackground() {  
 currentPlayer.run();  
 return null;  
 }  
  
 @Override  
 protected void done() {  
 SwingUtilities.*invokeLater*(LotoGame.this::updateBoard);  
 }  
 };  
  
 worker.execute();  
  
 if (board.isFull()) {  
 gameTimer.stop();  
 announceWinner();  
 }  
 }  
 });  
  
 gameTimer.start();  
  
 restartTimer = new Timer(1000, new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 restartPending = false;  
 restartTimer.stop();  
 resetGame();  
 }  
 });  
 }  
  
 private void updateBoard() {  
 for (int i = 0; i < board.getSize(); i++) {  
 LotoTile tile = board.getTile(i);  
 JButton button = (JButton) getContentPane().getComponent(i);  
 if (tile.getColor() != null) {  
 button.setBackground(tile.getColor());  
 button.setText(String.*valueOf*(tile.getNumber()));  
 button.setOpaque(true);  
 button.setBorderPainted(false);  
 }  
 else {  
 button.setBackground(null);  
 button.setText("");  
 button.setOpaque(false);  
 button.setBorderPainted(true);  
 }  
 }  
 }  
  
 private void announceWinner() {  
 int maxTilesPlaced = -1;  
 String winner = "";  
  
 for (Player player : players) {  
 int tilesPlaced = player.getTilesPlaced();  
 if (tilesPlaced > maxTilesPlaced) {  
 maxTilesPlaced = tilesPlaced;  
 winner = player.getName();  
 }  
 }  
  
 JOptionPane.*showMessageDialog*(  
 this, "Игра окончена! Победил " + winner + " игрок" + " и набрал " + maxTilesPlaced + " очков.",  
 "Победитель", JOptionPane.*INFORMATION\_MESSAGE*);  
 }  
  
 private void resetGame() {  
 board.clear();  
 for (Player player : players) {  
 player.reset();  
 }  
 gameTimer.start();  
 updateBoard();  
 }  
}

Файл LotoTile.java

@Builder  
class LotoTile {  
 @Getter  
 @Setter  
 private int number;  
 @Getter  
 @Setter  
 private Color color = null;  
}

Файл Board.java

@Builder  
class Board {  
 @Setter  
 private int sizeX;  
 @Setter  
 private int sizeY;  
 private final List<LotoTile> tiles = new ArrayList<>();  
  
  
 public int getSize() {  
 return sizeX \* sizeY;  
 }  
  
 public synchronized boolean placeTile(int tileNumber, Color color) {  
 for (LotoTile tile : tiles) {  
 if (tile.getNumber() == tileNumber && tile.getColor() == null) {  
 tile.setColor(color);  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public synchronized void addTile(LotoTile tile) {  
 tiles.add(tile);  
 }  
  
 public LotoTile getTile(int index) {  
 return tiles.get(index);  
 }  
  
 public boolean isFull() {  
 for (LotoTile tile : tiles) {  
 if (tile.getColor() == null) {  
 return false;  
 }  
 }  
 return true;  
 }  
  
 public void clear() {  
 for (LotoTile tile : tiles) {  
 tile.setColor(null);  
 }  
 }  
}

Файл Player.java

class Player implements Runnable {  
 @Getter  
 private String name;  
 private Color color;  
 private Board board;  
 @Getter  
 private int tilesPlaced;  
 private Random random;  
  
 public Player(String name, Color color, Board board) {  
 this.name = name;  
 this.color = color;  
 this.board = board;  
 this.tilesPlaced = 0;  
 this.random = new Random();  
 }  
  
 public boolean isFinished() {  
 return tilesPlaced >= board.getSize();  
 }  
  
 public void reset() {  
 tilesPlaced = 0;  
 }  
  
 @Override  
 public void run() {  
 int tileNumber;  
 boolean tilePlaced;  
 do {  
 tileNumber = random.nextInt(board.getSize()) + 1;  
 tilePlaced = board.placeTile(tileNumber, color);  
 } while (!tilePlaced);  
  
 tilesPlaced++;  
 System.*out*.println(name + ", фишка " + tileNumber);  
  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}