

Разработка игры «Сапёр» с графическим интерфейсом на языке Java в среде Apache Netbeans

Голованова Маргарита Антоновна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье описывается создание классической логической игры «Сапёр» на языке Java с графическим интерфейсом на базе библиотеки Swing. Программа реализует игровое поле с настраиваемыми размерами 9×9, 16×16, 16×30, автоматическую расстановку мин, подсчёт числа мин по соседству, рекурсивное открытие пустых областей, обработку левого и правого кликов мыши, таймер, счётчик оставшихся мин, а также переключение уровней сложности без дублирования компонентов интерфейса. Приложение демонстрирует событийно-ориентированное программирование, работу с двумерными массивами, использование JButton, MouseListener, рекурсивные алгоритмы обхода графа и управление игровым циклом.

Ключевые слова: Java, Swing, Сапёр, Minesweeper, графический интерфейс, рекурсия, обработка событий, игровое приложение.

Development of the “Minesweeper” game with a GUI in Java in the Apache Netbeans environment

Golovanova Margarita Antonovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The article describes the creation of the classic logic game "Minesweeper" in Java with a graphical interface based on the Swing library. The program implements a game board with customizable dimensions 9×9, 16×16, 16×30, automatic placement of mines, counting the number of mines in the neighborhood, recursively opening empty areas, processing left and right mouse clicks, timer, counter remaining mines, as well as switching difficulty levels without duplicating interface components. The application demonstrates event-oriented programming, working with two-dimensional arrays, using JButton, MouseListener, recursive graph traversal algorithms, and game loop control.

Keywords: Java, Swing, Minesweeper, graphical user interface, recursion, event handling, game application.

1 Введение

1.1 Актуальность

Игра «Сапёр» – одна из наиболее известных логических игр, входящая в стандартный набор операционных систем Windows. Её простое правило открывать все клетки, не содержащие мины, используя подсказки о количестве мин по соседству скрывает алгоритмические задачи: генерация случайного поля с гарантией отсутствия мины на первом ходу, рекурсивное вскрытие пустых областей для автоматического решения. Реализация «Сапёра» на Java с использованием Swing.

Библиотека Swing, входящая в стандартную поставку Java даёт возможность быстро создавать кроссплатформенные оконные приложения без установки дополнительных модулей, что важно для образовательных целей. Разработка «Сапёра» с поддержкой трёх уровней сложности, счётчика времени и оставшихся мин позволяет наглядно продемонстрировать различие между статическим интерфейсом и динамическими структурами данных.

1.2 Обзор исследований

В работе М.В. Юрлова [1] анализируются современные тенденции развития экосистемы Java, включая библиотеки для построения графического интерфейса. Отмечается, что Swing продолжает использоваться в образовательных проектах в силу своей полноты и кроссплатформенности, несмотря на появление JavaFX.

Учебное пособие А.В. Ермакова, М.С. Королёва, А.К. Кузьмина [2] посвящено основам разработки приложений на Java. Рассматриваются принципы объектно-ориентированного программирования, обработка событий, организация пользовательского интерфейса с помощью Swing. Данные темы непосредственно применимы к реализации логических игр, в том числе «Сапёра».

В статье Е.С. Петровой [3] описывается процесс создания интерактивной игры на Java с графическим интерфейсом. Особое внимание уделяется событийно-ориентированной модели, обработке кликов мыши, обновлению элементов интерфейса и управлению игровым циклом. Указанные механизмы необходимы для реализации таких компонентов «Сапёра», как открытие клеток, установка флагов и работа таймера.

Лабораторный практикум Е.Н. Шнейдерова, А.Ю. Писарчика, В.О. Казючича [4] содержит практические сведения о менеджерах компоновки (BorderLayout, GridLayout), создании меню (JMenuBar) и использовании javax.swing.Timer. Данные компоненты применяются для построения верхней панели управления и реализации динамической смены уровней сложности в разработанном приложении.

В работе А.Н. Бабкевича и А.А. Ерофеева [5] рассматривается создание игры на «чистой Java» без визуальных редакторов. Описываются алгоритмы расстановки мин, рекурсивного открытия пустых областей и корректного завершения игровой партии. Эти подходы были использованы для реализации методов revealCell() и calculateNeighbors() в представленном проекте.

Таким образом, в существующей литературе представлены фрагментарные сведения об отдельных аспектах создания игр на Java/Swing. Комплексная реализация «Сапёра», поддерживающая несколько уровней сложности, динамическое изменение размеров поля, таймер, счётчик оставшихся мин и переключение уровней без дублирования компонентов интерфейса, в доступных источниках отсутствует. Данное обстоятельство обосновывает актуальность и практическую значимость проведённого исследования.

1.3 Цель исследования

Цель работы разработать на языке Java интерактивную игру «Сапёр» с графическим интерфейсом на Swing, поддерживающую три уровня сложности новичок 9×9 , легкий 16×16 , средний 16×30 сложный, автоматический подсчёт мин по соседству, рекурсивное открытие пустых областей, обработку левого открытия и правого установка флага кликов, а также отображение таймера и количества оставшихся мин. Программа должна корректно завершать партию при подрыве на mine или победе, позволять начать новую игру без перезапуска приложения и, главное, корректно переключать уровни сложности без дублирования элементов интерфейса.

2 Материалы и методы

Для реализации использовался язык Java версии 11+ и встроенная библиотека Swing, включающая классы JFrame, JPanel, JButton, JLabel, JMenuBar, Timer, а также слушатели MouseListener. Генерация случайных мин выполнялась с помощью java.util.Random. Алгоритм определения числа мин по соседству основан на обходе восьми направлений. Рекурсивное открытие пустых областей реализовано через метод revealCell(), который при обнаружении клетки с нулевым числом соседних мин вызывает себя для всех смежных неоткрытых клеток.

Разработка велась в среде IntelliJ IDEA. Исходный код сохранялся в файл Minesweeper.java. Интерфейс построен с помощью менеджеров геометрии BorderLayout для верхней панели и GridLayout для игрового поля. Окно имеет динамический размер, вычисляемый на основе количества строк, столбцов и фиксированного размера кнопки 40×40 пикселей.

Ключевой методический приём, обеспечивающий корректную смену уровня в методе initGame() перед добавлением новых компонентов вызывается getContentPane().removeAll(), что удаляет все предыдущие панели и кнопки. Затем заново создаются верхняя панель (createTopPanel()) и поле. Это предотвращает накопление дублирующихся элементов.

3 Результаты

В данном разделе описывается процесс создания графической игры «Сапёр» на языке Java с использованием библиотеки Swing. Разработка велась в среде Apache NetBeans. Был создан файл Minesweeper.java (рис.1).

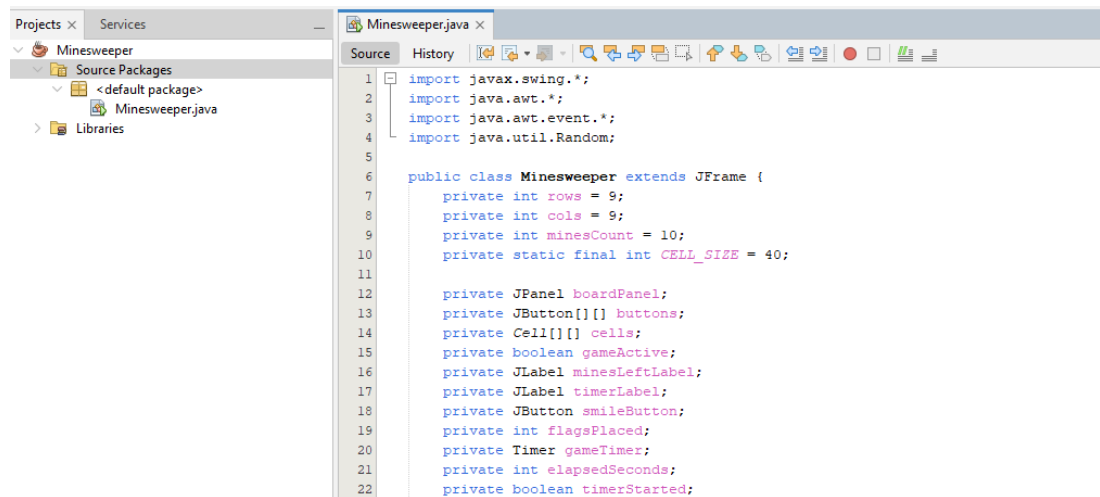


Рисунок 1. Интерфейс Apache NetBeans с открытым проектом Minesweeper

Импортирование библиотек и начало создания класса Minesweeper в Apache NetBeans (рис.2).

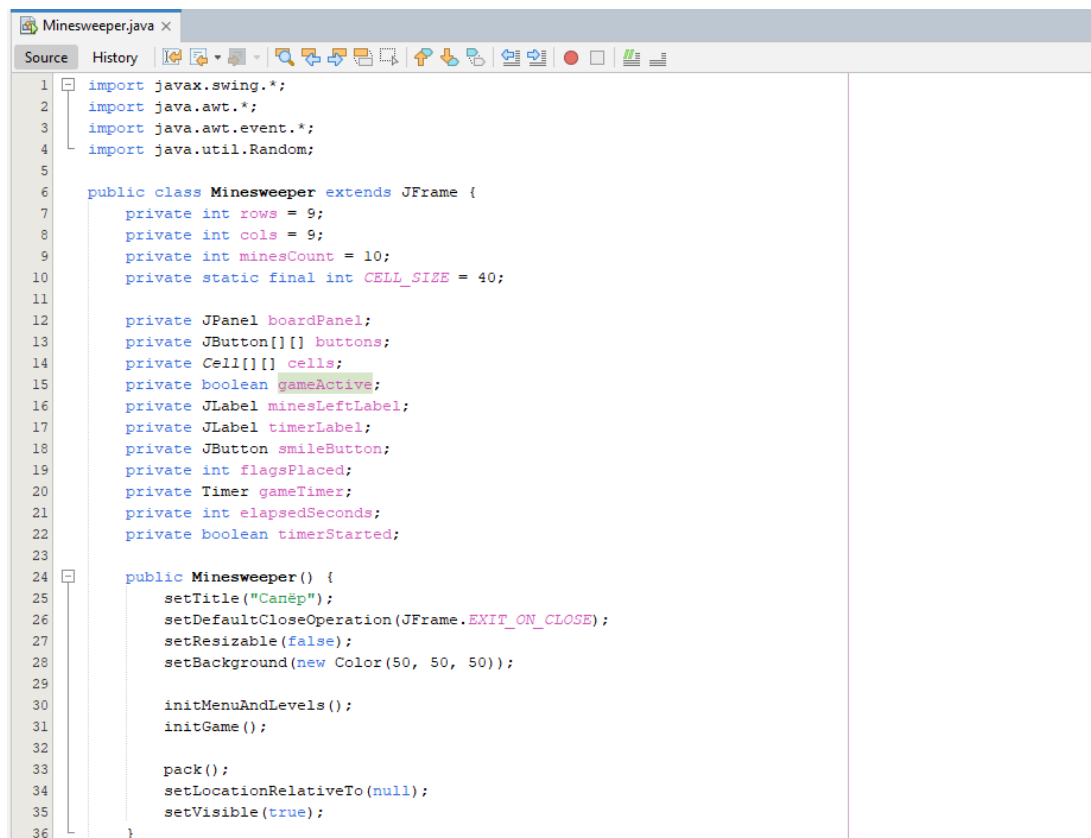


Рисунок 2. Импорт библиотек и начало класса Minesweeper в среде Apache NetBeans

Для хранения состояния каждой ячейки создан вложенный класс Cell с полями mine, revealed, flagged, neighborMines (рис.3).

```

401 private static class Cell {
402     private boolean mine;
403     private boolean revealed;
404     private boolean flagged;
405     private int neighborMines;
406
407     public boolean isMine() { return mine; }
408     public void setMine(boolean mine) { this.mine = mine; }
409     public boolean isRevealed() { return revealed; }
410     public void setRevealed(boolean revealed) { this.revealed = revealed; }
411     public boolean isFlagged() { return flagged; }
412     public void setFlagged(boolean flagged) { this.flagged = flagged; }
413     public int getNeighborMines() { return neighborMines; }
414     public void setNeighborMines(int neighborMines) { this.neighborMines = neighborMines; }
415 }

```

Рисунок 3. Класс Cell, хранящий состояние клетки

Метод `initGame()` создаёт двумерный массив `cells`, размещает мины методом `placeMines()`, вычисляет количество мин по соседству методом `calculateNeighbors()`. Затем создаётся панель `boardPanel` с сеткой `GridLayout`, на которой размещаются кнопки `JButton`. Каждая кнопка получает `MouseListener` для обработки левого и правого клика (рис.4).

```

73 private void initGame() {
74     gameActive = true;
75     timerStarted = false;
76     elapsedSeconds = 0;
77     flagsPlaced = 0;
78     if (gameTimer != null) gameTimer.stop();
79
80     cells = new Cell[rows][cols];
81     for (int i = 0; i < rows; i++) {
82         for (int j = 0; j < cols; j++) {
83             cells[i][j] = new Cell();
84         }
85     }
86     placeMines();
87     calculateNeighbors();
88
89     getContentPane().removeAll();
90
91     JPanel topPanel = createTopPanel();
92
93     boardPanel = new JPanel(new GridLayout(rows, cols, 1, 1));
94     boardPanel.setBackground(new Color(80, 80, 80));
95     boardPanel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
96
97     buttons = new JButton[rows][cols];
98     for (int i = 0; i < rows; i++) {
99         for (int j = 0; j < cols; j++) {
100             JButton button = createStyledButton();
101             final int row = i;
102             final int col = j;
103
104             button.addMouseListener(new MouseAdapter() {
105                 @Override
106                 public void mouseClicked(MouseEvent e) {
107                     if (!gameActive) return;
108                     if (!timerStarted) startTimer();
109                     if (SwingUtilities.isLeftMouseButton(e)) {
110                         onLeftClick(row, col);
111                     } else if (SwingUtilities.isRightMouseButton(e)) {
112                         onRightClick(row, col);
113                     }
114                 }
115             });
116         }
117     }
118 }

```

Рисунок 4. Инициализация игрового поля: создание массива клеток, расстановка мин и настройка слушателя мыши

Метод `placeMines()` случайным образом выбирает координаты и устанавливает флаг `mine = true`. Метод `calculateNeighbors()` для каждой не-мины подсчитывает количество мин в восьми соседних клетках (рис.5).

```

236 private void calculateNeighbors() {
237     int[] dr = {-1, -1, -1, 0, 0, 1, 1, 1};
238     int[] dc = {-1, 0, 1, -1, 1, -1, 0, 1};
239
240     for (int r = 0; r < rows; r++) {
241         for (int c = 0; c < cols; c++) {
242             if (cells[r][c].isMine()) continue;
243             int count = 0;
244             for (int d = 0; d < 8; d++) {
245                 int nr = r + dr[d];
246                 int nc = c + dc[d];
247                 if (nr >= 0 && nr < rows && nc >= 0 && nc < cols && cells[nr][nc].isMine()) {
248                     count++;
249                 }
250             }
251             cells[r][c].setNeighborMines(count);
252         }
253     }
254 }

```

Рисунок 5. Алгоритм расчёта соседних мин

При левом клике вызывается `onLeftClick()`, который проверяет мину и вызывает рекурсивное открытие `revealCell()`. При правом клике `onRightClick()` устанавливает или снимает флаг, обновляя счётчик (рис.6).

```

279 private void revealCell(int row, int col) {
280     Cell cell = cells[row][col];
281     if (cell.isRevealed() || cell.isFlagged()) return;
282
283     cell.setRevealed(true);
284     updateButton(row, col);
285
286     if (cell.getNeighborMines() == 0) {
287         int[] dr = {-1, -1, -1, 0, 0, 1, 1, 1};
288         int[] dc = {-1, 0, 1, -1, 1, -1, 0, 1};
289         for (int d = 0; d < 8; d++) {
290             int nr = row + dr[d];
291             int nc = col + dc[d];
292             if (nr >= 0 && nr < rows && nc >= 0 && nc < cols) {
293                 if (!cells[nr][nc].isRevealed() && !cells[nr][nc].isFlagged() && !cells[nr][nc].isMine()) {
294                     revealCell(nr, nc);
295                 }
296             }
297         }
298     }
299 }

```

Рисунок 6. Рекурсивное открытие пустых клеток

В верхней части окна добавлена панель меню с пунктами Beginner 9×9, 10 мин, Intermediate 16×16, 40 мин, Expert 16×30, 99 мин и New Game. При выборе уровня вызывается `setLevel()`, который обновляет размеры поля и перезапускает игру (рис.7).

```

38 private void initMenuAndLevels() {
39     JMenuBar menuBar = new JMenuBar();
40     menuBar.setBackground(new Color(70, 70, 70));
41     menuBar.setBorder(BorderFactory.createEmptyBorder(2, 5, 2, 5));
42
43     JMenu gameMenu = new JMenu("Игра");
44     gameMenu.setForeground(Color.WHITE);
45
46     JMenuItem beginnerItem = new JMenuItem("Легкий (9x9, 10 мин)");
47     JMenuItem intermediateItem = new JMenuItem("Средний (16x16, 40 мин)");
48     JMenuItem expertItem = new JMenuItem("Сложный (16x30, 99 мин)");
49     JMenuItem newGameItem = new JMenuItem("Новая игра (такой же уровень)");
50
51     beginnerItem.addActionListener(e -> setLevel(9, 9, 10));
52     intermediateItem.addActionListener(e -> setLevel(16, 16, 40));
53     expertItem.addActionListener(e -> setLevel(16, 30, 99));
54     newGameItem.addActionListener(e -> newGame());
55
56     gameMenu.add(beginnerItem);
57     gameMenu.add(intermediateItem);
58     gameMenu.add(expertItem);
59     gameMenu.addSeparator();
60     gameMenu.add(newGameItem);
61     menuBar.add(gameMenu);
62
63     setJMenuBar(menuBar);
64 }

```

Рисунок 7. Реализация меню выбора уровня сложности

Таймер реализован с помощью `javax.swing.Timer`, который каждую секунду увеличивает значение `elapsedSeconds` и обновляет метку. Счётчик оставшихся мин отображает разницу между общим числом мин и количеством установленных флагов (рис.8).

```
212 private void startTimer() {  
213     timerStarted = true;  
214     gameTimer = new Timer(1000, e -> {  
215         if (gameActive) {  
216             elapsedSeconds++;  
217             timerLabel.setText(String.format("%03d", elapsedSeconds));  
218         }  
219     });  
220     gameTimer.start();  
221 }
```

Рисунок 8. Реализация таймера

Программа позволяет комфортно играть на трёх уровнях сложности. На рисунках 8–16 показаны основные этапы игры.

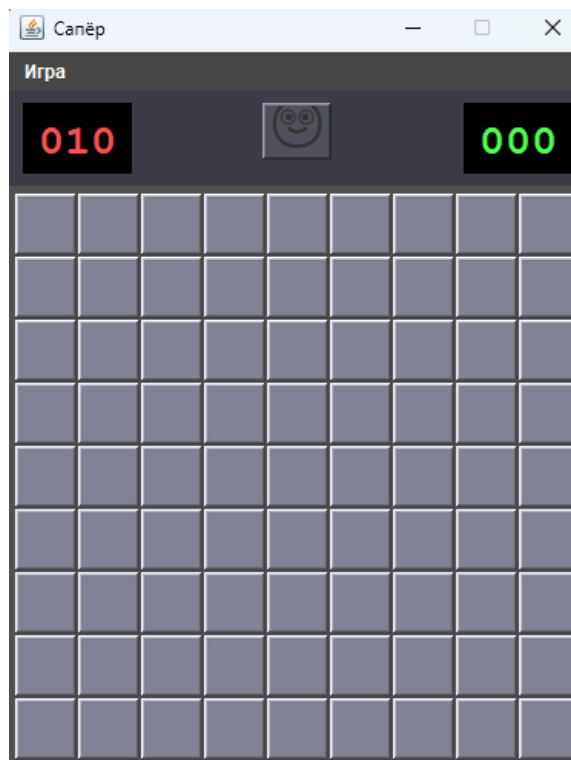


Рисунок 9. Начальный экран игры уровень Beginner. Поле 9×9, таймер 000, счётчик мин 010, все клетки закрыты



Рисунок 10. Меню выбора уровня сложности

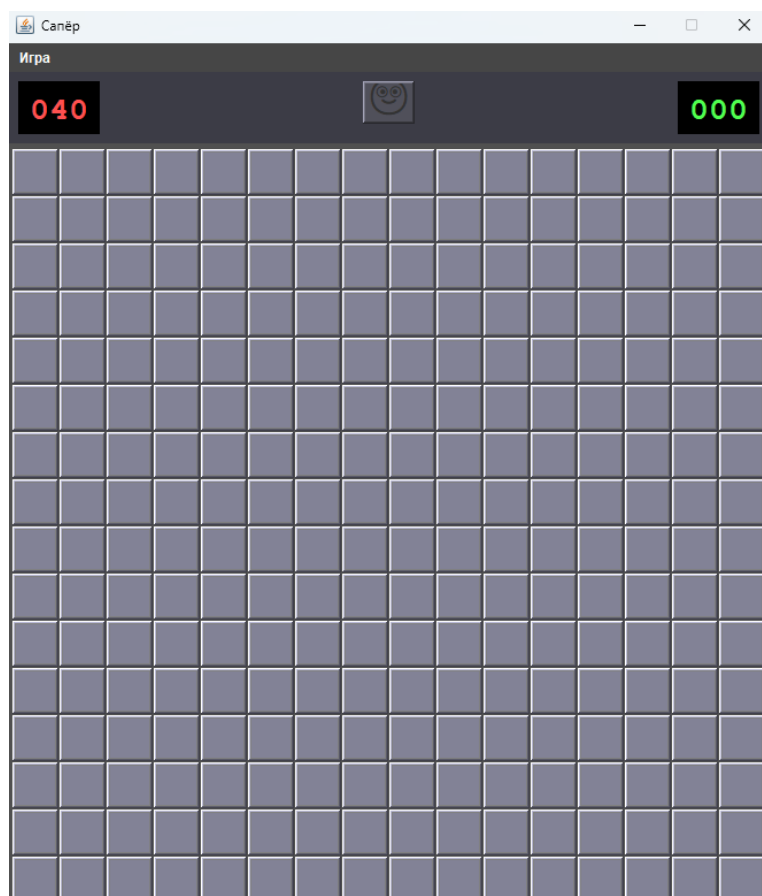


Рисунок 11. Панель интерфейса выбор уровня сложности Средний

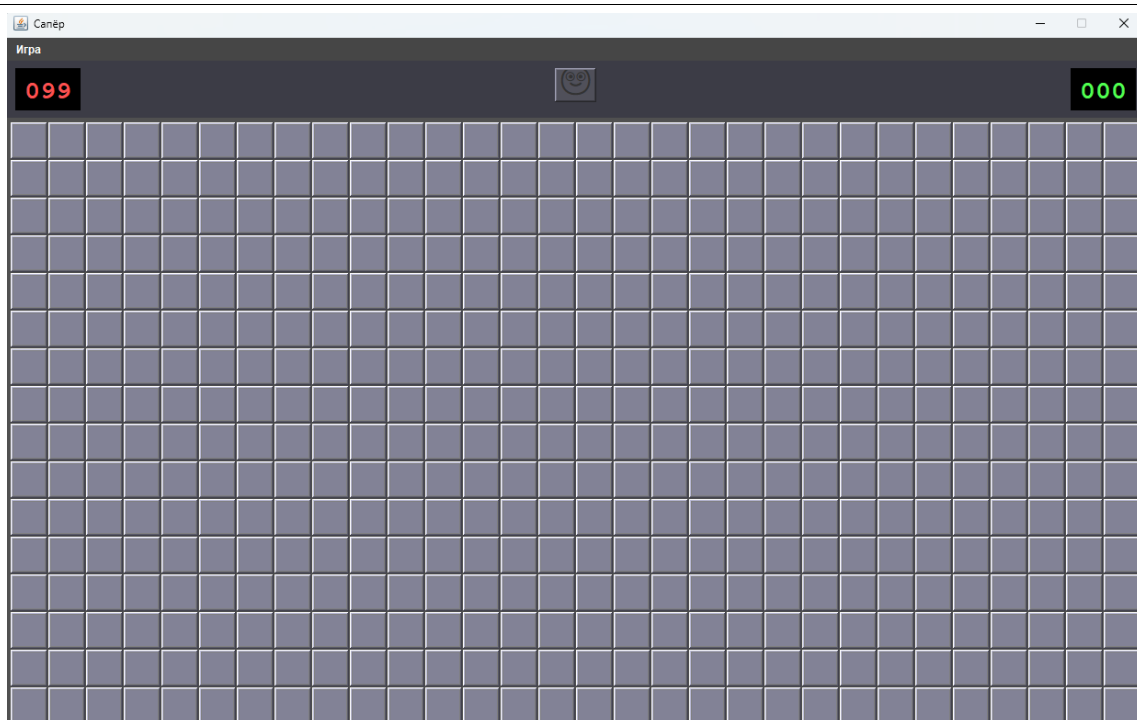


Рисунок 12. Панель интерфейса выбор уровня сложности Сложный

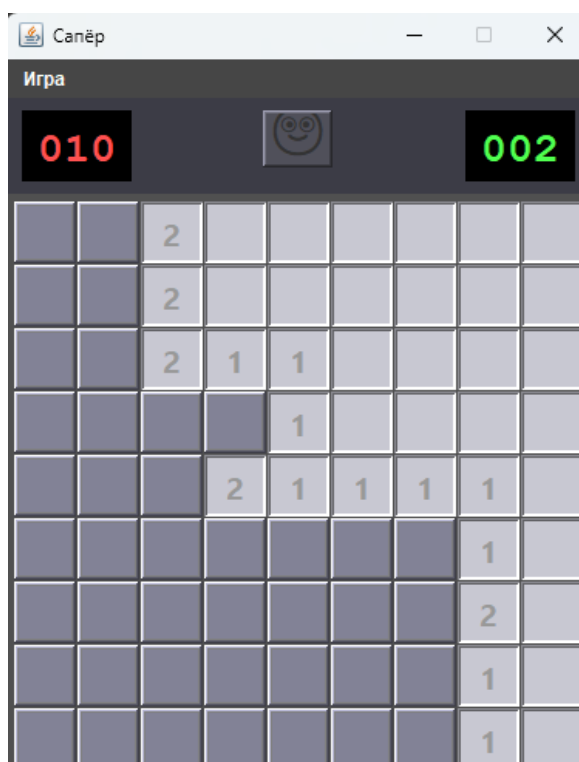


Рисунок 13. После открытия нескольких безопасных клеток отображаются цифры 1–3 или пустые клетки, рядом с которыми нет мин



Рисунок 14. Установка флагов на предполагаемые мины правой кнопкой мыши. Счётчик мин уменьшается

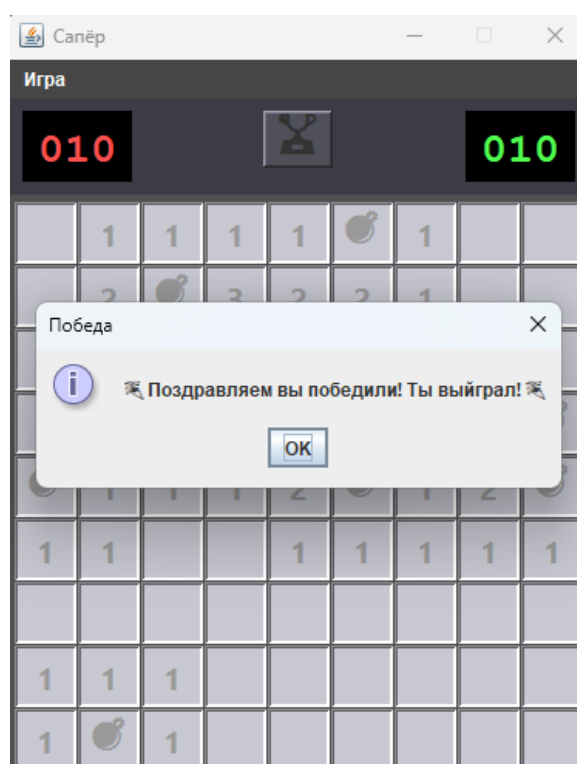


Рисунок 15. Сообщение о победе при открытии всех безопасных клеток

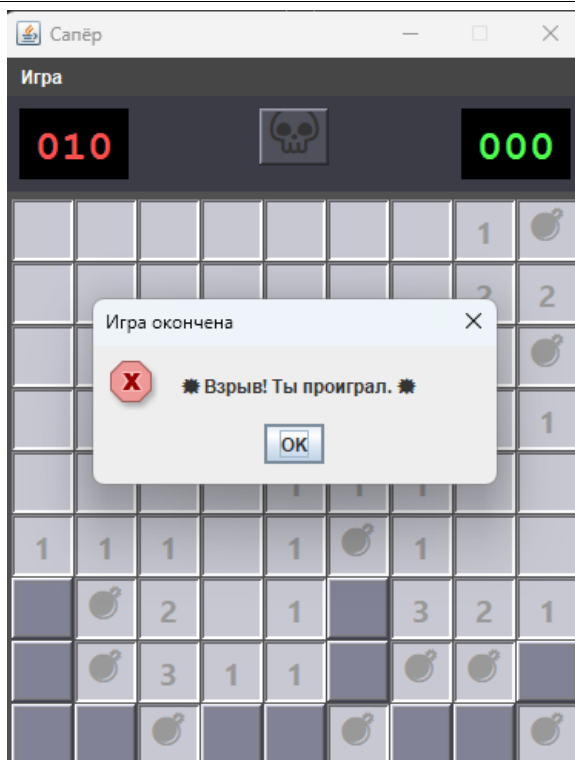


Рисунок 16. Сообщение о поражении при нажатии на мину. Все мины раскрываются, игровое поле блокируется

4 Выводы

В ходе выполненной работы спроектирована и реализована игра «Сапёр» на языке Java с использованием графического интерфейса Swing в среде Apache NetBeans. Разработанное приложение, по сути, воспроизводит ключевую механику классического «Сапёра», при этом предусматривая ряд функций, которые, вероятно, повышают удобство взаимодействия пользователя с программой. В частности, через меню доступен выбор уровня сложности лёгкий, средний и сложный, что позволяет адаптировать игровой процесс к разной подготовленности игрока.

Игровое поле поддерживает стандартные действия: открытие ячеек осуществляется левой кнопкой мыши, причём для пустых областей реализовано рекурсивное раскрытие соседних клеток. Маркировка предполагаемых мин выполняется правой кнопкой флаги можно как устанавливать, так и снимать, что делает управление более гибким. Дополнительно приложение отслеживает время прохождения с помощью таймера и отображает число оставшихся мин, тем самым предоставляя пользователю обратную связь о прогрессе. Наконец, по завершении партии выводятся информативные диалоговые окна, фиксирующие факт победы или поражения, достаточно очевидный, но важный элемент завершённости пользовательского сценария.

Библиографический список

1. Юрлов М.В. Тенденции в области разработок на Java на 2021 год // Научное сообщество студентов XXI столетия. Технические науки: сб. ст. по матер. ХСІХ студ. междунар. науч.-практ. конф. Новосибирск: Сибирская академическая книга, 2021. С. 78–80. URL: <https://elibrary.ru/item.asp?id=45687671>
2. Ермаков А.В., Королёв М.С., Кузьмин А.К. Разработка приложений на языке Java: учебное пособие. Саратов: СГТУ им. Гагарина Ю.А., 2024. 128 с. URL: <https://elibrary.ru/item.asp?id=90490447>
3. Петрова Е.С. Разработка пользовательского интерфейса и интерактивной игры на Java // Сборник статей РГРТУ им. В.Ф. Уткина. Рязань, 2025. С. 229–237. URL: <https://elibrary.ru/item.asp?id=82939536>
4. Шнейдеров Е.Н., Писарчик А.Ю., Казючич В.О. Разработка приложений на языке Java: лабораторный практикум. Минск: БГУИР, 2023. 92 с. URL: <https://elibrary.ru/item.asp?id=53971967>
5. Бабкевич А.Н., Ерофеев А.А. Разработка игры на чистой Java // Интернаука. 2025. № 11-1 (375). С. 15–16. URL: <https://elibrary.ru/item.asp?id=80532709>