МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Уральский государственный экономический университет»

(УрГЭУ)

КУРСОВАЯ РАБОТА

по дисциплине **«Объектно-ориентированное программирование»**

Тема: РАЗРАБОТКА ИГРЫ «2048»

Институт цифровых технологий управления и информационной безопасности

Направление подготовки
09.03.03 Прикладная информатика в экономике

Направленность (профиль)
Прикладная информатика в экономике

Кафедра
Информационных технологий и статистики

Студент
Бидаев Альфред Александрович

Группа *ПИЭ-21-1*

Руководитель
Панов Михаил Александрович
Кандидат экономических наук, доцент

Дата защиты: _____
Оценка: _____

Екатеринбург

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Объектно-ориентированное программирование, ООП – это одна из парадигм разработки, подразумевающая организацию программного кода, ориентируясь на данные и объекты, а не на функции и логические структуры.

Обычно объекты в подобном коде представляют собой полноценные блоки с данными, которые имеют определенный набор характеристик и возможностей. Объект может олицетворять что угодно – от человека с именем, фамилией, номером телефона, паролем и другой информацией до мелкой утилиты с минимумом характеристик из реального мира, но увеличенным набором функций. Объекты могут взаимодействовать друг с другом, пользователем и любыми другими компонентами программы.

Данная методология легла в основу немалого количества языков программирования, например: Java, C++, C#, Kotlin.

Актуальность выбранной темы для автора заключается в усовершенствовании и закреплении навыков программирования на языке Java. Пользователь в свою очередь получает продукт для изучения, а также возможность проведения досуга, используя созданный продукт.

Объектом исследования является игра «2048» как одна из игры, разрабатываемая на языке Java в среде IntelliJ IDEA.

Предметом исследования особенные характеристики игры «2048» и механизма её разработки.

Для достижения поставленной цели были поставлены следующие задачи:

1. Знакомство с понятиями предметной области.
2. Постановка технического задания
3. Применение на практике нужных методов изучаемых библиотек в программном коде

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Описание предметной области

Предметная область проекта определяет сферу деятельности, на которую нацелен проект. Это могут быть различные области: от программирования и IT-технологий до бизнеса и менеджмента. Каждый проект обладает своими особенностями, требующими специфических знаний и навыков.

Определение предметной области проекта необходимо для понимания того, кто будет использовать продукт, кому он будет предназначен и какие задачи будет решать. Например, если проект связан с программированием, нужно знать, какие языки программирования будут использоваться, какие функции должны быть включены в продукт и каким образом он будет взаимодействовать с другими программами.

В моем случае предметной областью является компьютерная игра «2048». Для разработки проекта использовался язык программирования Java и библиотеки Swing и AWT.

Графический интерфейс (GUI) – используется для отображения пользователю игрового поля, включающее в себя цветные плитки, расположенные в порядке 4x4.

Управление осуществляется за счет нажатия клавиш стрелок или WASD. Проигрышные ситуации определяются прописанными алгоритмами поражения.

1.2 Словарь предметной области

1.2.1 Библиотека Swing

Библиотека Swing была создана вслед за библиотекой AWT, и является её улучшенной версией. Swing — это, вероятно, самый продвинутый инструментарий на этой планете. Он обладает богатым набором виджетов, начиная от базовых виджетов, таких как кнопки, надписи, полосы прокрутки, и заканчивая более продвинутыми, такими как деревья и таблицы. Названия компонентов данной библиотеки начинается с буквы J, например, JFrame или JButton.

Основными компонентами библиотеки Swing являются:

— JInternalFrame – окно, существующее внутри другого окна верхнего уровня, например в JFrame.

— JProgressBar – строка, отображающая состояние протекание какого-либо процесса, например процесса загрузки.

— JTable – компонент, представляющий данные в виде таблицы.

— JTree – компонент, представляющий данные в иерархическом виде [4].

К особенностям библиотеки Swing можно отнести лёгкий вес, разнообразные элементы управления, широкие возможности настройки.

1.2.3 Библиотека Util

Java включает большое количество вспомогательных классов, широко используемых в других встроенных пакетах Java. Эти классы расположены в пакете java.util и используются для работы с набором объектов, взаимодействия с системными функциями низкого уровня, для работы с математическими функциями, генерации случайных чисел и манипуляций с датой и временем :

o Интерфейс Collection является основой всей иерархии классов-коллекций и определяет базовую функциональность любой коллекции - набор методов, которые позволяют добавлять, удалять, выбирать элементы коллекции. Классы, которые имплементируют интерфейс Collection, могут содержать дубликаты и пустые (null) значения. AbstractCollection, являясь абстрактным классом обеспечивает, служит основой для создания конкретных классов коллекций и содержит реализацию некоторых методов, определенных в интерфейсе Collection.

o Интерфейс Set расширяет интерфейс Collection. Классы, которые реализуют этот интерфейс не разрешают наличие дубликатов. В коллекции этого типа допускается наличие только одной ссылки типа null. Любой объект добавляемый в Set должен реализовать метод equals для того, чтобы его можно было сравнить с другими. AbstractSet, являясь абстрактным классом

представляет из себя основу для реализации различных вариантов интерфейса Set.

o Интерфейс List расширяет интерфейс Collection. Классы, которые реализуют этот интерфейс содержат упорядоченную последовательность объектов (Объекты хранятся в том порядке в котором они были добавлены). List обеспечивает также ListIterator, который позволяет перемещаться как вперед, так и назад, по элементам списка. AbstractList являясь абстрактным классом представляет из себя основу для реализации различных вариантов интерфейса List.

o Интерфейс Map не расширяет интерфейс Collection. Классы, реализующие этот интерфейс, хранят неупорядоченный набор объектов парами типа ключ/значение. Каждый ключ должен быть уникальным. Порядок следования пар ключ/значение не определен. AbstractMap являясь абстрактным классом представляет из себя основу для реализации различных вариантов интерфейса Map.

o Интерфейс SortedSet расширяет Set требуя, чтобы содержимое набора было упорядочено. Объект, имплементирующий SortedSet, содержит объекты, которые реализуют интерфейс Comparator или могут сравниваться с использованием внешнего объекта, реализующего интерфейс Comparator.

1.3 Анализ существующих решений

На данный момент существует большое количество различных вариантов исполнения этой игры, в том числе которые кардинально меняют концепцию игры, заменяя цифры на фигуры, увеличивая количество ячеек, добавляя челленджи для решения на скорость.

Среди наиболее популярных решений в App Store я нашел слудующие:

1. 2048: Number Puzzle Game. Разработчик Erkay Uzun. Это практически оригинальная версия игры за исключением возможности выбора размера поля и отмены последнего хода. В остальном это классическая версия 2048.

2. 2048. Разработчик Ketchapp. Данная версия выделяется ярким дизайном, челленджами и таблицей мирового рекорда за сутки, в остальном это классическая версия 2048.

3. 2048 – Number Puzzle Game. Разработчик Inspired Square FZE. Это уже кардинально отличающаяся по концепции игра. За основу механики взята «Три в ряд» и совмещено с игрой «2048»

Как итог, вариаций исполнения игры существует немало, многие похожи друг на друга, имеют свои достоинства и недостатки, и есть различные скрещения «2048» с другими жанрами игр. Разработчики любого уровня могут использовать эти решения как основа для собственного проекта.

1.4 Техническое задание

1.4.1 Функциональные требования

Функциональные требования определяют, каким должно быть поведение продукта в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи (пользовательские требования) в рамках бизнес-требований. Они являются одним из основых пунктов для любого технического задания разработки программного продукта.

К функциональным требованиям игры «2048» относится следующее:

— Отрисовка игрового окна
— Генерация ячеек для игрового поля
— Случайная генерация ячейки «2» или «4» после каждого хода, в случайной пустой ячейке
— Вывод результата игры
— Завершение игры если нет возможности хода
— Вывод на экран сообщения о проигрыше

1.4.2 Нефункциональные требования

Нефункциональные требования — требования, определяющие свойства, которые система должна демонстрировать, или ограничения, которые она должна соблюдать, не относящиеся к поведению системы. Например, производительность, удобство сопровождения, расширяемость, надежность, факторы эксплуатации. К таким требованиям могут относиться:

— Игра должна иметь простой и дружественный интерфейс
— Игра должна быть разработана с учетом максимальной доступности для пользователя, обеспечивая легкость освоения
— Игра должна обеспечивать бесперебойную работу без сбоев и ошибок

2 ПРОЕКТНАЯ ЧАСТЬ

2.1 Кратко об игре

Игровое поле состоит из сетки 4x4. Игра начинается. На сцене две плитки с номиналом 2. Передвигая, нужно сложить плитки одного «номинала». Движение возможно в 4 стороны. После каждого хода на свободной секции поля появляется новая плитка номиналом «2» или «4». Если после совершения хода на поле заняты все 16 полей и нет вариантов хода – игра заканчивается.

Причина популярности 2048 — простая механика. Освоить правила игры можно за 5–10 минут. Для этого не нужно читать многостраничные гайды. Низкий порог входа привлекает «казуальных» игроков, которые не любят излишнюю сложность в играх. К тому же это очень «залипательная» игра, которая захватывает внимание на долгие часы. Также игра 2048 развивает умственные способности: воображение, логическое мышление, стратегическое планирование, внимательность. Поэтому эту очень часто рекомендуют в качестве эффективного тренажера для быстрой «прокачки» мозга.

2.2 Проектирование и разработка классов

Ключевые классы и объекты, используемые в реализации игры:

1. Tile (Плитка): Представляет собой отдельную клетку на игровом поле. Имеет свойство «номинал», который может быть равен степени числа 2(2,4,8 и так далее).

2. Board (игровое поле): представляет собой игровое поле 4x4. Содержит массив плиток, которые образуют игровое поле.

3. Game (игра): управляет логикой игры. Отслеживает состояние игрового поля, перемещения плиток и их объединение. Определяет правила завершения игры.

Эти классы и объекты взаимодействуют между собой, обеспечивая функционирование игры.

2.3 Разбор классов программы

2.3.1 Класс Game

Это основной класс программы. В нем создается окно программы, рисуются плитки и проверяются нажатия клавиатуры. Программа начинает работу с метода main(), в котором вызывается функция setUpGUI(), которая создает экземпляр класса.

Метод paint() рисует игровое окно, в котором и будет происходить процесс игры.

Метод drawTiles() рисует на игровом поле новые плитки:

Имплиментированный метод keyPressed() из интерфейса KeyListener. Отвечает за управление клавиатурой и вызывает нужные методы при нажатии клавиш: WASD или стрелочек.

2.3.2 класс Board

В классе прописаны основные действия игры такие как: движение в четыре стороны, сложение одинаковых плиток, генерация новых плиток, завершение игры, и генерация игрового поля размером 4x4 плитки.

Конструктор класса Board: создает двумерный массив 4 на 4 ячейки

Метод getHighTile(): выдает наибольшую плитку на игровом поле

Метод spawn(): рандомно генерирует новую плитку с номиналом «2» в 80% или «4» в 20% случаев.

Метод gameOver():Проверяет, окончена ли игра - то есть проверяет, может ли какая-либо плитка (которая не равна 0) сочетаться с плитками рядом с ней - если нет, то игра окончена.

Методы движения up(), down(), right(), left(): вызываются при нажатии клавиш стрелок или WASD и вызывают методы horizontalMove() или verticalMove().

Методы horizontalMove() и verticalMove(): сравнивают значения двух плиток вместе, и если они совпадают или если одно из них равно 0 (обычная плитка) - их значения суммируются (при условии, что мы сравниваем две разные плитки, и они движутся в соответствующем направлении) - Используется рекурсия для прохождения всего столбца

2.3.3 Класс Tile

Это класс, который представляет собой одну ячейку, из которых строится игровое поле. Состоит из нескольких конструкторов, для плиток в разных моментах игры, таких как пустые плитки, и новые плитки с числом.

Содержит два конструктора: один конструктор для пустой ячейки, другой для ячейки с числом. Содержит так же геттеры и сеттеры: getValue() и setValue(int value), getColor() и setColor().

2.4 Описание алгоритмов и программных модулей

При проектировании классов игры мною были использованы только встроенные классы java и java swing.

Таб. 1. Структура классов игры 2048

Класс	Game	Board	Tile
Методы	main ()	spawn ()	getValue()
	setUpGUI ()	blackout()	setValue()
	paint ()	gameOver ()	setColor()
	keyListener()	up ()	getColor()
	drawTiles ()	down ()	
		left ()	
		right ()	

продолжение таблицы 1

		verticalMove()	
		horizontalMove()	
		getHighTile()	
		getScore()	
		getBoard()	

2.4.1 Алгоритмы игровые механики игры

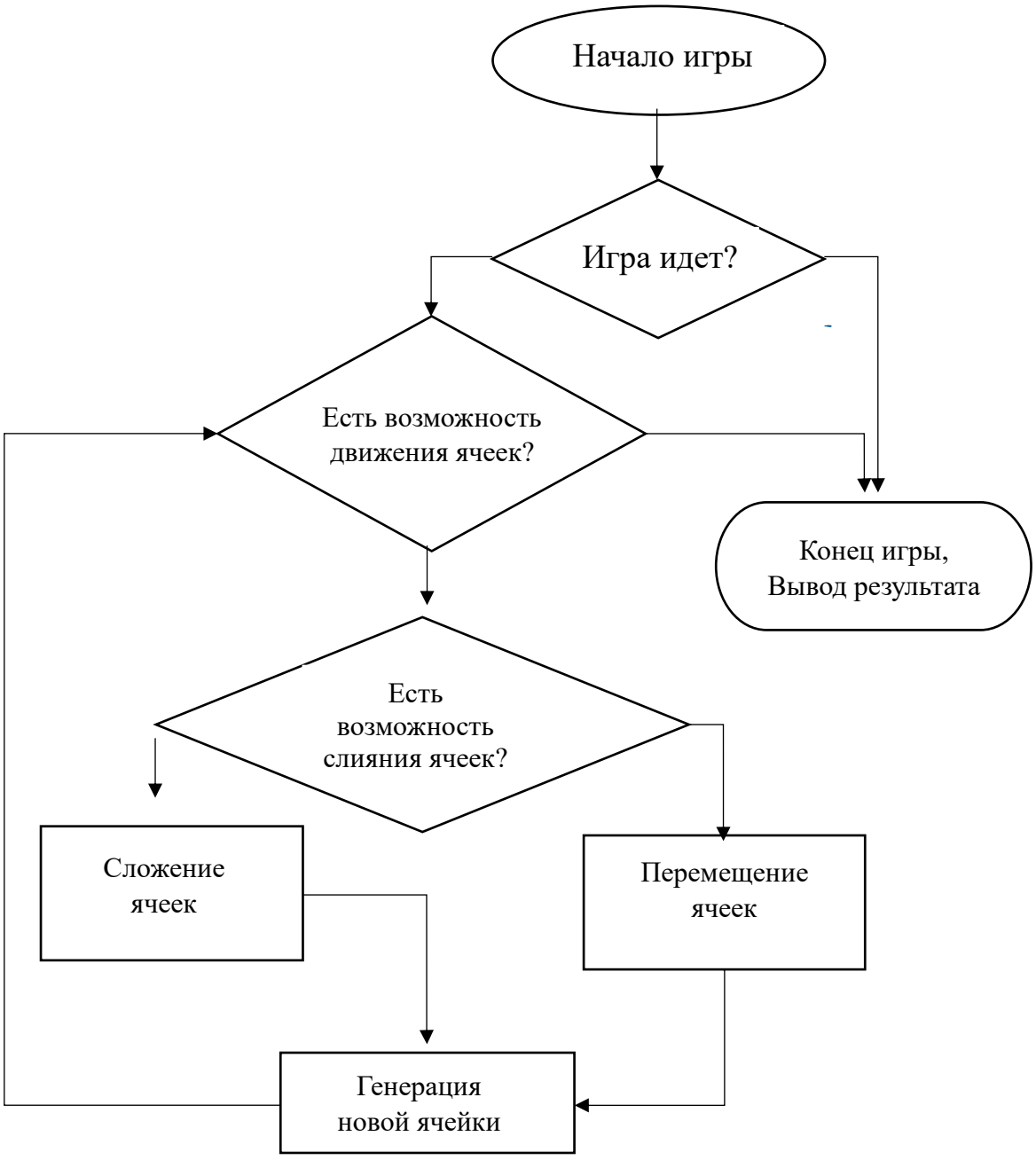


Рис. 1- Блок схема

Теперь рассмотрим подробнее игровую составляющую игры и ее механики. Из основных механик выделю:

- Генерацию поля
- Генерацию новых ячеек
- Сложение одинаковых ячеек
- Передвижение ячеек на поле
- Определение проигрышной ситуации

Для генерации игрового поля в конструкторе класса Board используется двумерный массив объектов класса Tile (ячейка). С помощью цикла «For» массив заполняется пустыми ячейками Tile.

```
public Board()
    {
        board = new Tile[4][4];
        for ( int i = 0; i < board.length; i++ )
        {
            for ( int j = 0; j < board[i].length; j++ )
            {
                board[i][j] = new Tile();
            }
        }
    }
```

Генерация новых ячеек происходит после нажатия пользователем клавиши и перемещения существующих ячеек. Выполняется это посредством вызова метода spawn() в имплементированном методе keyPressed(KeyEvent e):

```
@Override
    public void keyPressed( KeyEvent e )
    {
        if ( e.getKeyChar() == 'w' || e.getKeyCode() == KeyEvent.VK_UP )
```

```
{
    game.up();
    game.spawn();
    gameBoard = game.toString();
    frame.repaint();
}
```

Данный код выполняется при нажатии пользователем стрелочки вверх или клавиши «W», после чего сначала вызывается метод up(), обеспечивающий движение ячеек. Затем уже вызывается метод spawn(), который генерирует в свободной ячейке новую плитку. Новые ячейки генерируются размером «2» или «4», в случайно с вероятностью 80% на 20%, для чего применяется библиотека Math и метод random().

Код метода spawn():

```
public void spawn() {
    boolean empty = true;
    while ( empty )
    {
        int row = (int)( Math.random() * 4 );
        int col = (int)( Math.random() * 4 );
        double x = Math.random();
        if ( board[row][col].getValue() == 0)
        {
            if ( x < 0.2 )
            {
                board[row][col] = new Tile( 4 );
                empty = false;
            } else {
                board[row][col] = new Tile( 2 );
                empty = false;
```

```
            }
        }
    }
}
```

Для сложения ячеек используются методы verticalMove() и horizontalMove, которые в свою очередь вызываются при вызове методов up(), down(), right(), left().

Рассмотрим подробнее метод down() представляет собой часть логики перемещения элементов вниз в игровой доске. Он проходит по каждому столбцу (`i`) и, начиная снизу (`j` равно grids - 1`), проверяет значения ячеек. Если значение не равно 0, метод вызывает вспомогательный метод `verticalMove() для выполнения вертикального перемещения вниз.

Основная идея заключается в том, чтобы обеспечить, чтобы элементы "падали" вниз до самого нижнего пустого места в каждом столбце. Перед вызовом verticalMove() проверяется, находится ли текущая ячейка выше или на уровне переменной border. Если да, то выполняется вертикальное перемещение.

Листинг метода down():

```
public void down() {
    for ( int i = 0; i < grids; i++ ) {
        border = ( grids - 1 );
        for ( int j = grids - 1; j >= 0; j-- ) {
            if ( board[j][i].getValue() != 0 ){
                if ( border >= j ){ verticalMove( j, i, "down" ); }
            }
        }
    }
}
```

Метод verticalMove() представляет собой логику вертикального перемещения элементов на игровой доске, вызываемую методом down() для каждого столбца. Вот краткое описание его работы:

1. initial и compare представляют собой две ячейки, где initial - самая нижняя занятая ячейка в столбце, а compare - ячейка, которую мы рассматриваем для вертикального перемещения.

2. Если значение initial равно 0 или равно значению compare, и row больше border (что гарантирует, что мы перемещаемся вниз), то происходит объединение двух ячеек: их значения суммируются, и значение compare становится 0. Если initial имеет значение 0, то добавляется соответствующий балл к переменной score.

3. Если условие объединения не выполняется, то border уменьшается (если направление `"down"`) или увеличивается (если направление не "down"). Затем метод вызывает сам себя рекурсивно для следующего уровня.

Листинг метода verticalMove():

```
    private void verticalMove( int row, int col, String direction ){
        Tile initial = board[border][col];
        Tile compare = board[row][col];
        if ( initial.getValue() == 0 || initial.getValue() == compare.getValue() ){
            if ( row > border || ( direction.equals( "down" ) && ( row < border ) ) ){
                int addScore = initial.getValue() + compare.getValue();
                if ( initial.getValue() != 0 ){
                    score += addScore;
                }
                initial.setValue( addScore );
                compare.setValue( 0 );
            }
        } else {
```

```
        if ( direction.equals( "down" ) ){
            border--;
        }else{
            border++;
        }
        verticalMove( row, col, direction );
    }
```

Метод gameOver проверяет, завершена ли игра в соответствии с определенными условиями. Он анализирует содержимое игровой доски (board), представленной двумерным массивом. Игра считается завершенной, если для каждой ячейки на доске нет возможности объединения с соседними ячейками по определенным правилам.

В данном случае count увеличивается каждый раз, когда условия для ячейки не выполняются. Если после проверки всех ячеек count достигает значения 16 (размер доски), метод возвращает true, указывая на завершение игры. В противном случае возвращается false. Это, вероятно, используется для определения, находится ли игра в состоянии, когда больше невозможно сделать ходы.

Одна из проверок на проигрыш в методе gameOver():

```
public boolean gameOver(){
    int count = 0;
    for ( int i = 0; i < board.length; i++ ){
        for ( int j = 0; j < board[i].length; j++ ){
            if ( board[i][j].getValue() > 0 ){
                if ( i == 0 && j == 0 ){
                    if ( board[i][j].getValue() != board[i + 1][j].getValue()
                            && board[i][j].getValue() != board[i][j + 1].getValue() )
                    { count++; }}
```

2.5 Тестирование программного комплекса

2.5.1 Описание методики тестирования

Существует множество различных типов тестирования, каждый из которых имеет свои особенности и преимущества. Рассмотрим функциональное тестирование.

Функциональное тестирование — это такое тестирование, когда программное обеспечение проверяют на соответствие функциональным требованиям и спецификациям.

Функциональные требования — это то, чего заказчик ожидает от ПО. Функциональные требования определяют, что должна делать система и без чего она не будет работать. Функциональные требования обычно указывают в документе спецификации, или техническом задании на разработку.

2.5.2 Проверка тестов и их анализ

Проведем функциональное тестирование программы. Проверим правильность поведения программы при нажатии клавиш, проверим правильные ли методы используются в зависимости от выбранного действия и проверим корректность отрисовки ячеек.

Первым тестом проверяем запуск программы и запуск всех конструкторов:

Результат теста на рисунке 2 стр. 20.

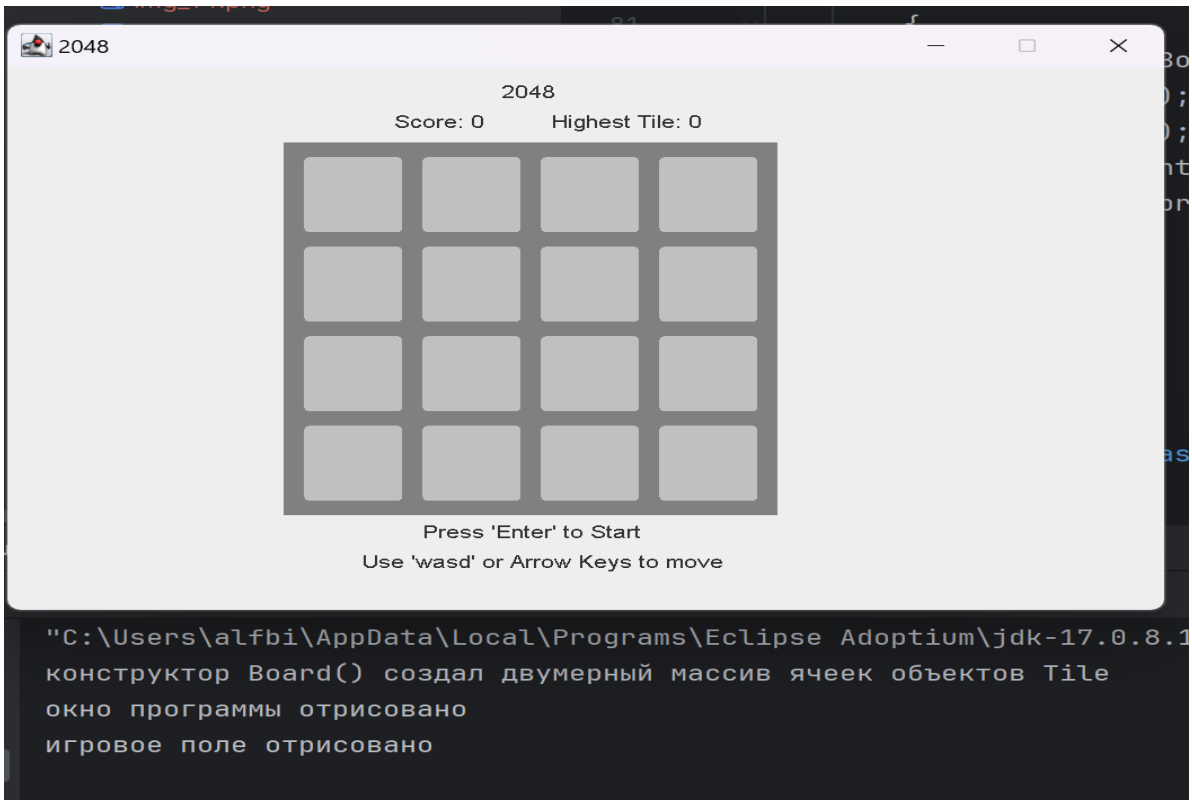


Рис. 2 – Тест запуска программы

Следующий тест проверяет правильность действий при нажатии клавиш движения плиток. Результат на рисунке 3.

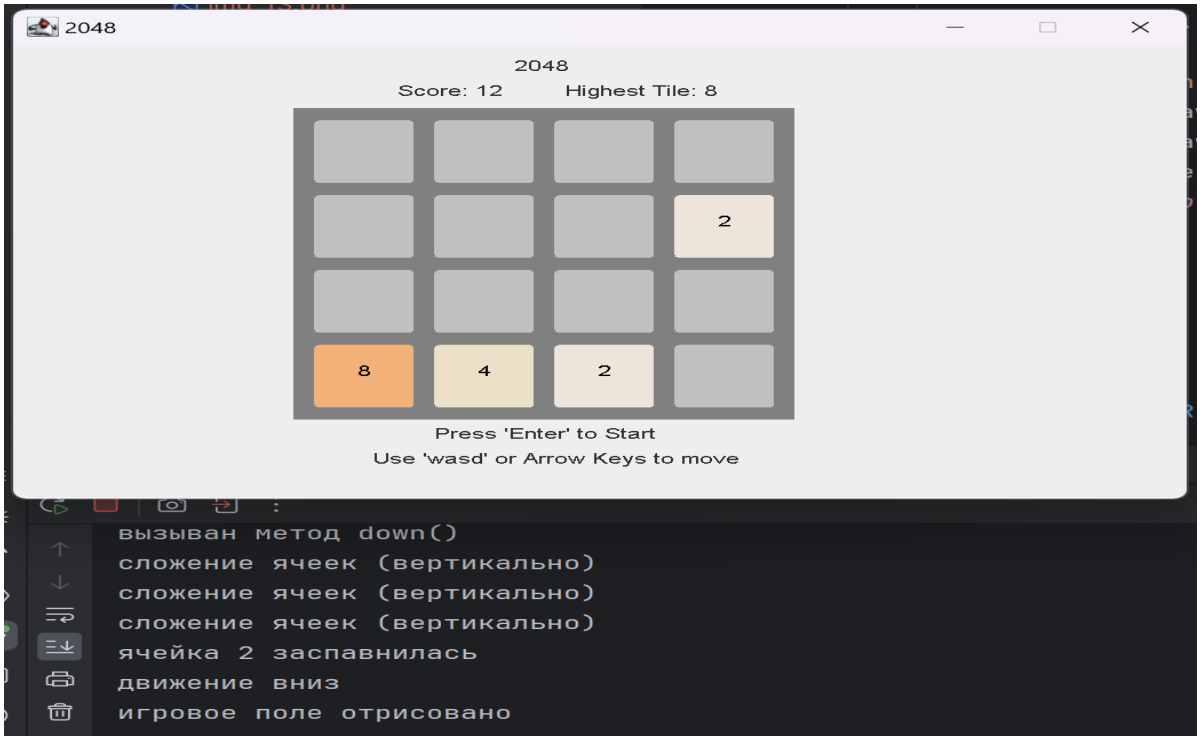
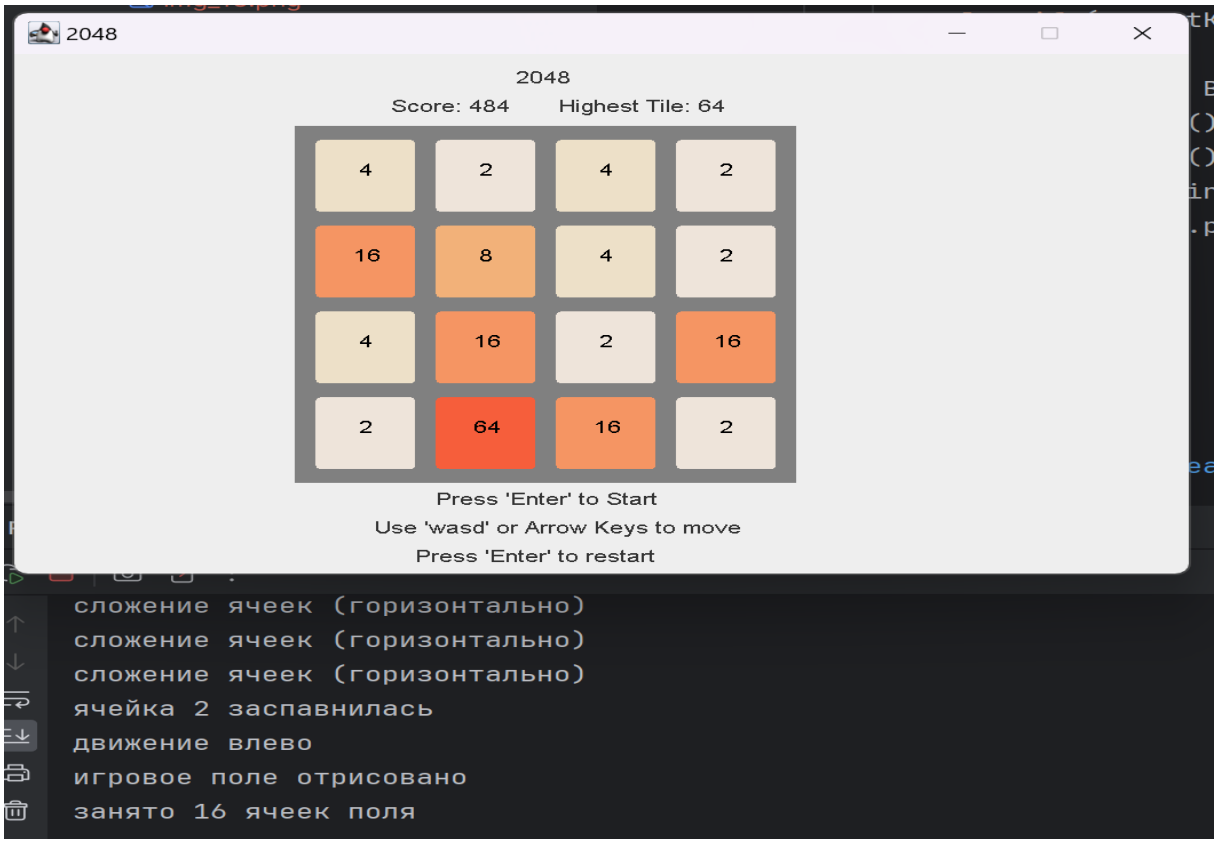


Рис 3. – тест движения плиток

Тест на определение полного игрового поля и проигрышной ситуации. Результат теста на рисунке 4.



Тест на корректную генерацию новых ячеек игрового поля. Результат теста на рисунке 4.

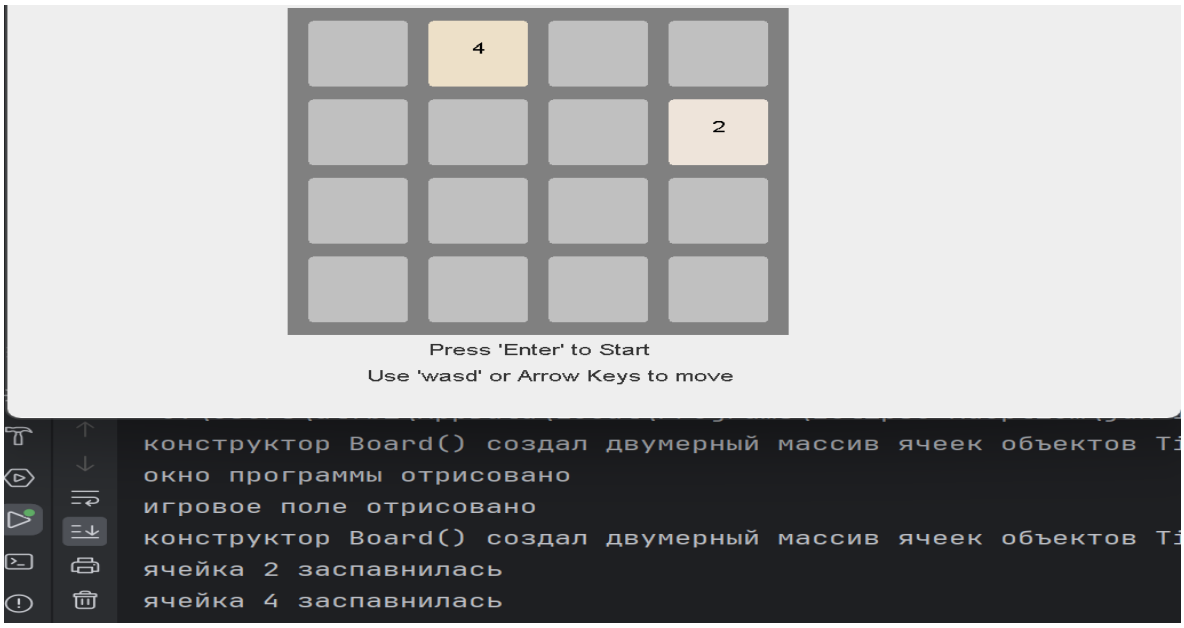


Рис.4 – тест корректной генерации ячеек

2.6 Руководство пользователя

Руководство пользователя — документ, назначение которого — предоставить людям помощь в использовании некоторой системы. Документ входит в состав технической документации на систему и, как правило, подготавливается техническим писателем.

Данное руководство пользователя предназначено для игры 2048, придуманной Габриэллом Чирулли. Эта версия игры написана мной на языке Java для отработки практических навыков.

Цель игры – получить ячейку с числом 2048.

Как играть:

- Для управления в игре используются клавиши WASD и клавиши стрелок.
- Для запуска новой игры нажмите клавишу Enter.
- Для перезапуска уже начатой игры или при проигрыше нажмите Enter.

Советы для игроков:

Не спешите. В данной игре важно продумывать каждый ход, прежде чем его совершать, не стоит передвигать ячейки в беспорядке туда-сюда.

Просчитывайте ходы. Этот момент нужно обязательно занести в правила игры "2048", поскольку, если его не соблюдать, вероятность выигрыша ничтожно мала.

В одно движение необходимо соединять как можно больше ячеек. Интересно то, что в одно движение вы можете соединить даже три цифры. Это позволит вам быстро расчистить поле и собрать больше крупных цифр.

Обращайте больше внимания на маленькие цифры. Вы допустите большую ошибку, если будете собирать много крупных цифр, при этом не следя за маленькими.

ЗАКЛЮЧЕНИЕ

В результате моей работы над проектом, связанным с изучением объектно-ориентированного программирования и разработкой игры "2048" на языке Java в среде IntelliJ IDEA, я успешно углубил свои навыки программирования. Актуальность темы проявилась не только в усовершенствовании моих навыков, но и в создании продукта, который не только подходит для обучения, но и предоставляет пользователям возможность интересного досуга.

В ходе исследования особенностей игры "2048" и разработки её механизма, я успешно выполнил поставленные задачи. Знакомство с понятиями предметной области позволило глубже понять основы ООП. Постановка технического задания была выполнена в соответствии с требованиями, а применение методов изучаемых библиотек в программном коде подчеркнуло практическую направленность проекта.

Таким образом, разработанная программа полностью соответствует поставленным требованиям, обеспечивая не только обучение, но и приятное времяпрепровождение для пользователей.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Библиотека Swing // Все о Java и SQL. - URL: https://java-online.ru/libs-swing.xhtml.
2. Блог Web Программиста. [Статья] / «Что такое 2048 и почему эти игры так популярны?» URL: http://juice-health.ru/raznoe/933-chto-takoe-2048-i-pochemu-eti-igry-tak-populyarny
3. Implementing a 2048 Solver in Java [статья] - URL: https://www.baeldung.com/2048-java-solver
4. https://timeweb.com/ru/community/articles/obektno-orientirovannoe-programmirovanie
5. psk-group [электронный ресурс] – режим доступа: свободный - URL: https://psk-group.su/znacheniya/cto-takoe-predmetnaya-oblast-proekta
6. Kathy Sierra, Bert Bates Head First Java [Text] / Kathy Sierra, Bert Bates—: O'Reilly Media, Incorporated, 2005 — 688 p.
7. Tproger [электронный ресурс]: https://tproger.ru/articles/vyjavlenie-i-sbor-trebovanij-k-po-ultimate-guide
8. Как проводить юзабилити-тестирование [статья] – URL: https://media.contented.ru/opyt/instrukcii/kak-provodit-yuzabiliti-testirovanie/
9. Функциональное тестирование ПО [электронный ресурс]: режим доступа: свободный – URL: https://tquality.ru/blog/tipiurovniim/
10. betacode. [Электронный ресурс]: – Режим доступа: URL: https://betacode.net/13527/java-inputstream, свободный.
11. Машнин Т. Графические интерфейсы пользователя Java [Текст] / Издательские решения 2019. – 760 с.
12. Хорстманн К. Java. Библиотека профессионала [Текст] / Хорстманн К. — 1-й том. —: Диалектика-Вильямс, 2020 — 864 с.
13. Wikipedia. Юзабилити-тестирование. - URL: https://ru.wikipedia.org/wiki/Юзабилити-тестирование

Приложение А

Исходный код программы на GitHub:
https://github.com/Wrdalf/OOP_labs/tree/main/labs/2048GameProject/src/main