Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
федерального государственного бюджетного образовательного учреждения высшего образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»**
(МИ ВлГУ)

Факультет информационных технологий и радиоэлектроники
Кафедра информационных систем

КУРСОВАЯ РАБОТА

по курсу Прикладная разработка на Java

на тему: Разработка игры “Wordle”

Руководитель

к. т. н., доц. каф. ИС
(уч. степень, звание)

Метёлкин А. С.
(фамилия, инициалы)

(оценка)

(подпись) (дата)

Члены комиссии

Студент ИС - 122
(группа)

Клинцов С. М.
(фамилия, инициалы)

(подпись) (Ф.И.О.)

(подпись) (Ф.И.О.)

(подпись) (дата)

Муром 2025

В ходе курсовой работе было разработано десктоп-приложение “Wordle”:

- Спроектирована и реализована логика игры;
- Разработан интерфейс взаимодействия с пользователем;
- Проведено тестирование и отладка проекта.

СОДЕРЖАНИЕ

Введение

В современном мире цифровые игры стали не только популярным способом развлечения, но и эффективным инструментом для развития когнитивных навыков, таких как логическое мышление, память и концентрация. Одной из таких игр является Wordle — словесная головоломка, в которой игроку необходимо угадать загаданное слово за ограниченное число попыток. Простота правил в сочетании с увлекательным геймплеем сделали эту игру крайне популярной среди пользователей разных возрастов.

Объектом исследования данной курсовой работы является разработка собственной версии игры Wordle на языке Java.

Цель работы — закрепление знаний и применение навыков, полученных во время прохождения курса прикладной разработки на Java в рамках реального приложения с применением паттернов проектирования.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучение правил и механик оригинальной игры Wordle;
2. Анализ возможных подходов к реализации игры на Java;
3. Проектирование архитектуры приложения;
4. Разработка графического интерфейса (Swing/JavaFX);
5. Реализация логики игры, включая генерацию слов, проверку ввода, подсветка введённых букв;
6. Реализация мультиязычного интерфейса;
7. Тестирование и отладка приложения;
8. Оптимизация кода и улучшение пользовательского опыта.

Результатом выполнения всех поставленных задач является готовое приложение, способное удовлетворить изначальным требованиям.

1 Анализ технического задания

Исходным заданием работы является разработка собственного приложения-игры Wordle на языке Java. Основными трудностями такой работы является создание дизайна пользовательского интерфейса, написание логики игры и продумывание UX. Решение этих проблем позволит построить грамотный план работы для дальнейшего равномерного достижения проектом состояния готовности.

1.1 Выбор технологии разработки приложения

При разработке игры Wordle на Java необходимо было выбрать подходящие инструменты и технологии, обеспечивающие эффективную сборку проекта, удобный графический интерфейс и корректную работу с XML-данными, для реализации смены языка. Рассмотрим обоснование выбора каждой из используемых технологий.

1.1.1 Система сборки: Maven

Для автоматизации сборки проекта, управления зависимостями и настройки жизненного цикла разработки использовался Apache Maven.

Сравнение с Gradle:

- Gradle использует Groovy/Kotlin DSL, что делает его более гибким и производительным за счёт инкрементальных сборок.
- Maven работает на основе XML-конфигурации (pom.xml), что делает его более строгим и предсказуемым, но менее гибким.
- В отличие от Gradle, Maven имеет более простую структуру конфигурации и широкую поддержку в IDE (IntelliJ IDEA, Eclipse).

Причины выбора Maven:

- В предыдущем курсовом проекте мной использовался Gradle, поэтому для разнообразия и закрепления навыков было решено попробовать Maven.
- Maven предоставляет удобный механизм управления зависимостями и стандартизированную структуру проекта.
- Поддержка плагинов (maven-compiler-plugin, maven-shade-plugin) упрощает сборку и упаковку приложения.

1.1.2 Графический интерфейс: JavaFX

Для реализации пользовательского интерфейса был выбран JavaFX, а не классический Swing.

Сравнение со Swing:

- Swing — устаревшая, но стабильная библиотека, встроенная в Java SE.
  - Плюсы: лёгкость в освоении, хорошая документация.
  - Минусы: устаревший дизайн, сложность в создании современных анимаций.
- JavaFX — более современный фреймворк, поддерживающий:
  - CSS-стилизацию, FXML для декларативного описания интерфейса.
  - Анимации, 3D-графику и мультиплатформенность.
  - Лучшую производительность благодаря аппаратному ускорению.

Причиной выбора JavaFX стали:

- Более современный и гибкий подход к разработке UI.
- Возможность стилизации через CSS, что упрощает создание визуально привлекательного интерфейса.
- Поддержка FXML, что позволяет разделять логику и разметку (по аналогии с веб-разработкой).

1.1.3 Работа с XML: Jakarta XML Binding (JAXB)

Для динамической смены языка использовалась библиотека JAXB. Эта библиотека позволяет легко сериализовать Java-объекты в XML и обратно через аннотации (@XmlRootElement, @XmlAttribute).

В Java 11+ стандартный JAXB был удалён из JDK, поэтому используется реализация от Jakarta EE.

Альтернативы (например, Jackson XML или SimpleXML) требуют дополнительных зависимостей и сложнее в настройке для простых задач.

1.1.4 Среда разработки: VScode

Средою разработки был выбран редактор Visual Studio Code, бесплатно распространяющийся компанией Microsoft. Он обладает поддержкой большого количества языков программирования, подсветкой синтаксиса, имеет встроенный терминал и возможность расширения функционала за счёт установки дополнений, также присутствуют инструменты для отладки кода и возможность работы с сервисами по контролю версий. Всё вышеперечисленное делает эту среду одной из лучших и удобных для Java разработки.

2. Проектирование приложения

После анализа требований и возможных архитектурных решений была выбрана следующая структура проекта (см. Рисунок 1). Основной упор сделан на модульность, разделение логики и интерфейса, а также гибкость для возможного расширения функционала.

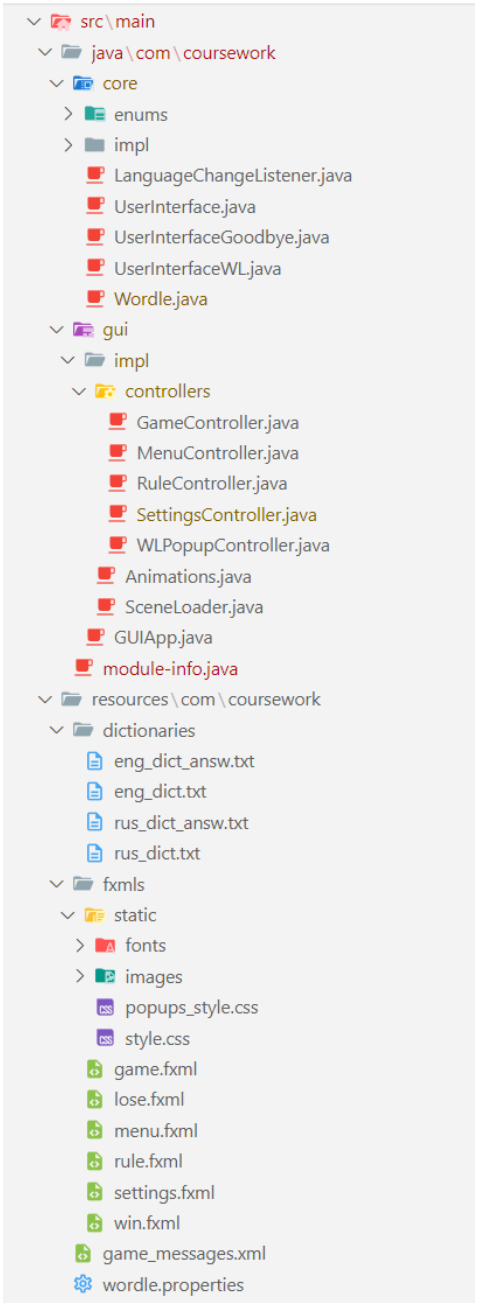

Рисунок 1 – Структура проекта

2.1 Абстрактный класс Wordle

Было принято решение начать проектирование с создания абстрактного класса Wordle, который содержит основной интерфейс игры, который в дальнейшем станет шаблоном для конкретных реализаций игры “Wordle”.

Класс Wordle предоставляет:

- Инициализацию игры:
  - Загрузку настроек (Settings)
  - Выбор случайного слова-ответа (answer) из словаря
  - Установку количества попыток (attempts) и длины слова (wordLength)
- Логику сравнения слов (comparingWords):
  - Принимает введенное пользователем слово и сравнивает его с загаданным
  - Возвращает массив строк с цветовыми метками:
    - Зеленый - буква на правильной позиции
    - Желтый - буква есть в слове, но в другой позиции
    - Без цвета - буквы нет в слове
- Абстрактные методы для реализации в конкретных версиях игры:
  - submitWord - обработка введенного слова
  - onWordSubmitted - реакция на корректно введенное слово
  - onInvalidWord - реакция на некорректное слово
  - onGameOver - обработка окончания игры
  - restartGame - перезапуск игры

2.2 Интерфейсы UserInterface, UserInterfaceGoodbye, UserInterfaceWL

Для обеспечения гибкости и модульности пользовательского интерфейса были разработаны три ключевых интерфейса, каждый из которых отвечает за определенный аспект взаимодействия с игроком.

1) UserInterface - Базовый интерфейс взаимодействия, который определяет основные экраны приложения:

```
package com.coursework.core;
import java.io.IOException;
public interface UserInterface<T> {
    T mainMenu() throws IOException;
    T settings() throws IOException;
    T gameplay(String ... keyword) throws IOException;
}
```

2) UserInterfaceGoodbye - Интерфейс завершения сеанса. Отвечает за финальное взаимодействие при завершении работы приложения:

```
package com.coursework.core;

import java.io.IOException;

public interface UserInterfaceGoodbye<T> {
    T goodbye() throws IOException;
}
```

3) UserInterfaceWL - Интерфейс обработки результатов игры:

```
package com.coursework.core;

import java.io.IOException;

public interface UserInterfaceWL<T> {

    T win() throws IOException;

    T lose() throws IOException;

}
```

2.3 Интерфейсы LanguageChangeListener и Localizable

Для реализации многоязычной поддержки в игре была разработана система локализации, состоящая из двух ключевых компонентов:

- LanguageChangeListener - интерфейс обратного вызова, позволяющий компонентам UI реагировать на смену языка в реальном времени:

```
package com.coursework.core;

import com.coursework.core.enums.Languages;

public interface LanguageChangeListener {
    void onLanguageChanged(Languages newLanguage);
}
```

При изменении языка все зарегистрированные слушатели автоматически получают уведомление через метод onLanguageChanged().

- Интерфейс Localizable, который предоставляет на реализацию метод void updateText(), для инициализации смены языка у конкретного окна.

Такая система позволит инкапсулировать логику реализации настроек языка и логику самой игры.

3 Разработка приложения

Далее приступаем к реализации задуманной структуры и наших интерфейсов.

3.1 Интерфейс взаимодействия с пользователем

Для разработки был выбран JavaFX в связке с FXML – декларативным языком разметки на основе XML. Такой подход обеспечивает четкое разделение логики (контроллеры) и визуального представления (FXML-макеты). FXML позволяет описывать структуру интерфейса в виде иерархии компонентов, их свойства (размеры, стили, расположение) и привязки к данным, что делает код более наглядным и удобным для редактирования. Стилизация выполняется через CSS-файлы, что обеспечивает единообразие дизайна и простую настройку внешнего вида.

Для получения доступа к визуального представления, реализуем нужные интерфейсы из п. 2.2: UserInterface<Scene>, UserInterfaceWL<FXMLLoader>.

Их имплементацией будет заниматься класс SceneLoader. Принцип реализации одинаковый, поэтому рассмотрим на примере метода mainMenu(), который возвращает сцену для меню:

```
        public Scene mainMenu() throws IOException {
              FXMLLoader loader = new
FXMLLoader(getClass().getResource("/com/coursework/fxmls/menu.fxml"));
              Parent root = loader.load();

              MenuController controller = loader.getController();
              controller.init(languageManager, this);

              return new Scene(root);
          }
```

Этот класс является единственным инкапсулированным доступом к реализации gui-сцен в приложении.

3.2 Контроллеры MenuController, GameController, SettingsController, RuleController, WLPopupController.

Для реализации логики взаимодействия между интерфейсом и ядром игры были разработаны специализированные контроллеры, каждый из которых отвечает за свою часть функционала:

- MenuController - управление главным меню приложения.
- GameController – реализует абстрактный класс Wordle и предоставляет управление основным игровым процессом.
- SettingsController - управление настройками приложения.
- RuleController - отображение правил игры.
- WLPopupController - управление всплывающими окнами результатов.

Каждый из этих контроллеров имплементирует интерфейс Localizable, открывая возможность локализации.

3.3 Класс Settings

Далее, не менее важным классом стал Settings. Он реализует централизованное управление настройками игры Wordle, используя паттерн Singleton (гарантирует единственный экземпляр на всё приложение).

Его ключевые функции:

1) Управление конфигурацией

- Загрузка настроек из файла wordle.properties:

  – При первом запуске использует встроенный файл из ресурсов (DEFAULT_PROPERTIES_FILE_PATH).

  – Для последующих запусков — файл в OS-специфичной директории:

    - Windows: %LOCALAPPDATA%\Wordle\

- Linux: ~/.config/wordle/

- Сохранение изменений в тот же файл.

2) Хранение основных параметров

- Язык игры.
- Длина слова.
- Количество попыток.

3) Работа со словарями

Автоматическая загрузка словарей в зависимости от языка:

- answerDictionary — словарь слов, которые могут быть загаданы.
- wordDictionary — словарь всех допустимых слов для ввода.

4) Поддержка смены языка

Реализует паттерн Слушатель (через интерфейс LanguageChangeListener):

- При смене языка обновляет словари.
- Оповещает подписчиков (например, UI) через notifyLanguageChanged().

5) Доступ к настройкам

Предоставляет геттеры для всех параметров

3.4 Класс LanguageManager

LanguageManager - центральный класс системы локализации, применяющий паттерн Registry (для централизованного управления объектами или сервисами, которые могут быть запрошены по ключу). Класс реализует:

- Загрузку и хранение локализованных текстов из XML-файла
- Управление словарями слов для разных языков

- Механизм динамического обновления текстовых элементов интерфейса
- Гибкую систему провайдеров текста с возможностью расширения

Он хранит в себе все ссылки на классы реализации интерфейса Localizable:

```
    private final Map<String, Localizable> localizableComponents = new
HashMap<>();
```

Также он является прослушиваемым классом в Settings через реализацию интерфейса LanguageChangeListener лямбда-функцией:

```
    private LanguageManager() {
            settings = Settings.getInstance();
            settings.addLanguageChangeListener(newLanguage ->
                localizableComponents.values().forEach(component ->
    component.updateText(this)));
          ...
```

Эта система обеспечивает:

- Простую интеграцию новых языков
- Автоматическое обновление интерфейса при смене языка
- Единую точку доступа ко всем локализованным ресурсам
- Поддержку как статических текстов, так и динамически формируемых сообщений

Использование JAXB для работы с XML позволяет легко модифицировать текстовые ресурсы без изменения кода приложения, что значительно упрощает процесс локализации и дальнейшей поддержки проекта.

3.5 Класс Dictionary

Класс Dictionary представляет собой реализацию словаря слов, который используется для хранения и обработки слов в приложении. Основные функции класса включают получение случайного слова из словаря и проверку наличия слова в словаре.

- Инициализация:

- Словарь инициализируется путем к файлу, содержащему слова (в формате .txt)
- Путь передается в конструктор и сохраняется в поле path

- Методы:
  - getRandomWord() - возвращает случайное слово из словаря
    - Сначала подсчитывает общее количество строк (слов) в файле
    - Затем генерирует случайное число в диапазоне от 1 до количества строк
    - Возвращает слово, соответствующее сгенерированной строке
  - isTheWordInTheDictionary() - проверяет наличие слова в словаре
    - Последовательно читает файл построчно
    - Возвращает true, если слово найдено, и false в противном случае

- Хранение данных:
  - Слова хранятся в обычных текстовых файлах (.txt)
  - Каждое слово располагается на отдельной строке

4 Тестирование

Тестирование программного обеспечения (ПО) является неотъемлемой частью процесса разработки и обеспечивает высокое качество конечного продукта. Тестирование выполняется для проверки соответствия ПО требованиям, выявления ошибок, дефектов и недоработок, а также для проверки функциональности, производительности и безопасности.

4.1 Тестирование меню

1) Запуск игры

   Ожидаемый результат: инициализация проигрывания анимации, отображение текста, соответствующего выбранному языку.

   Результат: соответствует ожидаемому.

2) Нажатие на кнопку “играть”

   Ожидаемый результат: переход на окно игры.

   Результат: соответствует ожидаемому.

3) Нажатие на кнопку “как играть”

   Ожидаемый результат: появление модального окна с правилами.

   Результат: соответствует ожидаемому.

4) Нажатие на кнопку “языки”

   Ожидаемый результат: появление модального окна с выбором языка.

   Результат: соответствует ожидаемому.

5) Нажатие на кнопку “выход”

   Ожидаемый результат: выход из приложения.

   Результат: соответствует ожидаемому.

4.2 Тестирование смена языка

1) Нажатие на одну из кнопок смены языка:

Ожидаемый результат: Смена языка интерфейса, смена словарей в самой игре.

Результат: соответствует ожидаемому.

4.3 Тестирование игры

1) Нажатие на кнопку “в меню”:

   Ожидаемый результат: возврат в меню.

   Результат: соответствует ожидаемому.

2) Ввод букв через физическую клавиатуру:

   Ожидаемый результат: вводимые буквы появляются на поле.

   Результат: соответствует ожидаемому.

3) Ввод букв через виртуальную клавиатуру:

   Ожидаемый результат: вводимые буквы появляются на поле.

   Результат: соответствует ожидаемому.

4) Нажатие клавиши BACK SPACE на физической клавиатуре или клавиши “Удалить” на виртуальной:

   Ожидаемый результат: удаление буквы с поля.

   Результат: соответствует ожидаемому.

5) Нажатие клавиши ENTER на физической клавиатуре или клавиши “Ввод” на виртуальной, при условии, что на поле были введены все 5 букв слова:

   Ожидаемый результат: слово есть – его сохранение на поле и подсветка букв. Слова нет – очистка поля ввода слова, проигрывание анимации “покачивания”.

   Результат: соответствует ожидаемому.

6) Реакция на ввод ответного слова:

   Ожидаемый результат: появление модального окна с поздравлениями.

   Результат: соответствует ожидаемому.

7) Реакция на окончание количества доступных попыток угадывания слова:

Ожидаемый результат: появление модального окна с сообщением о проигрыше.

Результат: соответствует ожидаемому.

4.4 Тестирование окон проигрыша/победы

1) Нажатие на кнопку “играть снова”:

Ожидаемый результат: обнуление результатов предыдущей игры.

Результат: соответствует ожидаемому.

2) Нажатие на кнопку “в меню”:

Ожидаемый результат: возврат в меню.

Результат: соответствует ожидаемому.

4.5 Функциональное тестирование словарей

1) Проверка на корректность работы функции isTheWordInTheDictionary.

Ожидаемые результаты:

- Возвращает true, если слово есть в словаре;
- Возвращает false, если слово отсутствует в словаре;
- Возвращает false, если слова не существует в словаре;
- Возвращает false, если регистр букв в слове не соответствует.

Результат: соответствует ожидаемому.

В результате все тесты прошли без каких-либо ошибок, все функции выполнялись штатно и их результат можно было предугадать заранее.

Заключение

В результате выполнения курсовой работы была разработана игра Wordle на Java, реализующая все основные механики оригинальной головоломки. В ходе работы достигнуты поставленные цели:

Реализована базовая игровая логика – генерация слов, проверка ввода, подсчет попыток и определение победных/проигрышных условий. Создан удобный пользовательский интерфейс с использованием JavaFX и FXML, обеспечивающий интуитивное взаимодействие. Добавлена поддержка локализации через LanguageManager, позволяющая легко расширять список языков. Организована модульная структура проекта, что упрощает дальнейшее развитие и поддержку кода.

Разработанное приложение успешно выполняет свою основную функцию – предоставляет увлекательный и адаптируемый игровой процесс. Оно может быть использовано как обучающий пример для изучения Java, JavaFX и ООП, как и в качестве базы для более сложных модификаций (добавление новых режимов, мультиплеера, статистики).

В ходе работы были закреплены навыки по работе с Java, примененены паттерны проектирования, а также освоена настройка сборки проекта через Maven.

Таким образом, поставленные задачи выполнены в полном объеме, а полученный опыт будет полезен для более сложных проектов в будущем.

Список литературы и электронные ресурсы

1. Шилдт, Герберт Java. Полное руководство: 10-е изд. : Пер. с англ. – СПб. : ООО “Диалектика”, 2020 – 1488 с.
2. Б. Эванс, Д. Кларк, М. Фербург Java для опытных разработчиков: 2-е изд. : Пер. с англ. – СПб. : 2024 – 736 с.
3. Документация по JavaFX [электронный ресурс]: Режим доступа: https://openjfx.io/openjfx-docs.
4. GitHub с исходным кодом программы [электронный ресурс]: Режим доступа: https://github.com/unfamiliarS/IS-122-Java-Klincov-Wordle.