|  | **Технически университет - Варна**  **Факултет по изчислителна техника и автоматизация**  **Катедра “Софтуерни и интернет технологии”** |
| --- | --- |

**Курсов проект - ООП 1**

**Тема: Приложение за работа с електронни таблици**

| **Изготвил:** | **Проверил:** |
| --- | --- |
| **Сечкин Салим Рахим** |  |
| **Фак. № 23621624** |  |
| **Спец.: Софтуерни и интернет технологии** |  |

**Варна**

**2025**

1. **Увод**

**1.1. Описание и идея на проекта**

Проектът е Java-базирано конзолно приложение за работа с електронни таблици (CSV формат). То позволява зареждане, редактиране и запазване на таблици чрез команден ред. Поддържа различни типове данни (цяло, дробно число, текст) и използва Command Pattern за CLI за по-добра структура и разширяемост без switch-case.

**1.2. Цел и задачи за разработка**

Основната цел на разработката е създаването на функционално приложение за работа с електронни таблици в команден ред, което демонстрира прилагането на принципите на обектно-ориентираното програмиране. Програмата трябва да бъде в състояние да чете и записва текстови файлове, съдържащи таблични данни, както и да позволява манипулации с клетките от таблицата чрез команди.

### **Задачи на разработката**

* Разработване на архитектура на приложението
* Имплементация на основни типове клетки:
* Целочислени стойности
* Дробни числа
* Текстови стойности
* Формулни клетки
* Създаване на система за обработка на команди
* Имплементация на файлови операции
* Добавяне на поддръжка за формули и изчисления
* Тестване и валидация на функционалността

**1.3. Структура на документацията**

Документацията е организирана в следните основни раздели:

Увод - описание на проекта и неговите цели

Преглед на предметната област - основни концепции и изисквания

Проектиране - архитектура и структура на проекта

Реализация - детайли за имплементацията

Тестване - тестови сценарии и примери

Заключение - обобщение на постигнатите резултати

1. **Преглед на предметната област**

**2.1. Основни концепции и алгоритми, които ще бъдат използвани**

Разработката на проекта се базира на следните основни концепции и техники от обектно-ориентираното програмиране и структури от данни:

* **Абстракция и наследяване** – чрез абстрактния клас Cell и неговите подтипове (IntegerCell, DoubleCell, StringCell), които позволяват представяне на различни типове данни в таблицата.
* **Полиморфизъм** – използва се при работата с обекти от тип Cell, където реалната функционалност се определя в зависимост от конкретния подтип.
* **Command Pattern** – приложен е за обработка на потребителските команди в командния ред, което позволява лесно добавяне и поддържане на нови команди без използване на switch-case.
* **Работа с текстови файлове** – програмата използва BufferedReader и BufferedWriter за зареждане и записване на данни във файлове с CSV-подобен формат.
* **Разделяне на отговорности (Separation of Concerns)** – логиката на програмата е разделена на ясно обособени класове, като: управление на таблицата, команди, клетки и команден интерфейс.
* **Валидация на вход** – при редактиране на клетки се извършва проверка дали стойността отговаря на допустим тип (цяло число, дробно число, низ).
* **Индексиране и визуализация на таблицата** – таблицата се представя като двумерен списък от клетки, който се отпечатва с подравняване чрез форматиране на текст.

**2.2. Подходи, методи за решаване на поставените задачи (евентуално модели и стандарти)**

За решаването на поставените задачи в проекта е избран модулен и обектно-ориентиран подход, който позволява логическо разделяне на функционалността, по-лесна поддръжка и бъдещо разширяване. Използвани са следните методи и добри практики:

* Обектно-ориентирано моделиране – чрез абстрактен клас Cell и негови конкретни имплементации се постига гъвкавост при представяне на различни видове данни в таблицата.
* Command Pattern – използван като заместител на switch-case, с цел по-чиста архитектура. Всяка команда е имплементирана в отделен клас, който имплементира общ интерфейс Command.
* Обработка на команди от потребителя – реализирана чрез речник (Map<String, Command>), който асоциира всяка команда с конкретния ѝ клас. Това улеснява добавянето на нови команди.
* Стандартизирано форматиране и четене на файлове – използват се Java класовете BufferedReader и BufferedWriter за работа с текстови файлове. Форматът на таблицата е подобен на CSV, което позволява лесен анализ и запис.
* Валидиране и обработка на потребителски вход – входните стойности се проверяват чрез регулярни изрази и подходящо парсване, за да се определи типът на клетката (int, double, String).
* Стандарти за кодиране и документация – в проекта се използва JavaDoc за генериране на автоматична документация, както и конвенционално именуване на класове и методи според Java стандартите.

**2.3. Функционални изисквания (потребителски изисквания, права, роли, статуси, диаграми и качествени (нефункционални) изисквания)**

* **Отваряне на файл** чрез команда open <път>, който съдържа таблични данни в текстов формат.
* **Извеждане на таблицата** на екрана с команда print, с добре подравнени колони.
* **Редактиране на клетки** чрез команда edit <ред> <колона> <стойност>, където стойността може да бъде цяло число, дробно число или текстов низ.
* **Записване на таблицата** чрез save (в същия файл) или saveas <път> (в нов файл).
* **Затваряне на таблицата** чрез команда close.
* **Помощна информация** чрез команда help.
* **Изход от програмата** чрез exit.  
  **Обработка на невалидни команди и аргументи**, чрез съобщения за грешка, без прекъсване на програмата.
* Няма роли, потребителски профили – приложението се използва от един потребител в команден ред.
* Статусът на приложението се променя динамично според отворения или затворения файл.
* Диаграми - няма.

### **Качествени (нефункционални) изисквания**

* **Изпълнимост** – програмата трябва да работи стабилно при нормални и гранични случаи на вход.
* **Устойчивост при грешки** – невалиден вход не трябва да води до срив.
* **Лесна поддръжка и разширяемост** – чрез Command Pattern нови команди могат лесно да се добавят.
* **Четимост на кода** – спазени са именувания и структура според добри Java практики

1. **Проектиране**

**3.1. Обща структура на проекта (пакети, които ще се реализират)**

#### **core**

Съдържа основните класове, описващи таблицата и нейните клетки:

* Spreadsheet – представя електронната таблица.
* Cell – абстрактен клас за клетка.
* IntegerCell, DoubleCell, StringCell – конкретни имплементации на Cell.

#### **commands**

Съдържа всички команди, достъпни през CLI. Всяка команда е реализирана като отделен клас:

* Command – интерфейс.
* OpenCommand, EditCommand, PrintCommand, SaveCommand, SaveAsCommand, CloseCommand, HelpCommand, ExitCommand.

#### **cli**

Отговаря за обработката на потребителския вход:

* CommandProcessor – анализира и изпълнява командите на потребителя.
* Може да съдържа и InputHandler, ако се използва допълнителна логика за въвеждане.
* Main – клас с main() метод, който стартира приложението.

### **Описания на класовете:**

### **Spreadsheet**

Основният клас, който управлява електронната таблица. Съдържа двумерен списък от клетки (Cell), предоставя методи за зареждане от файл, запис, редактиране и отпечатване на съдържанието. Отговаря за основната логика при работа с таблични данни.

### **Cell (абстрактен клас)**

Представлява абстракция на една клетка от таблицата. Всеки наследник трябва да имплементира методите getDisplay() и getValue(). Служи като основа за различни типове данни.

### **IntegerCell, DoubleCell, StringCell**

Конкретни имплементации на Cell за съхранение съответно на цели числа, дробни числа и текстови низове. Всеки клас съдържа логика за форматиране и интерпретация на съответния тип.

### **Command (интерфейс)**

Определя поведението на всички команди, достъпни в командния ред. Всеки клас, който имплементира този интерфейс, трябва да има метод execute(String args).

### **OpenCommand, SaveCommand, SaveAsCommand, PrintCommand, EditCommand, CloseCommand, HelpCommand, ExitCommand**

Класове, които реализират конкретни действия на потребителя чрез командния ред. Всеки от тях имплементира интерфейса Command.

### **CommandProcessor**

Отговаря за разпознаване и изпълнение на потребителските команди. Съдържа речник (Map<String, Command>) с всички регистрирани команди и логика за извикване на съответната команда според входа.

### **Main**

Съдържа началната точка на програмата – метода main(). Стартира приложението, инициализира Spreadsheet и CommandProcessor, и започва да приема команди от потребителя.

1. **Реализация**

**4.1. Реализация на класове (с JavaDoc - включва реализацията на функционалностите с**

**техните класове и малки фрагменти от кода за важните функционалности)**

* Cell:

/\*\*

\* Абстрактен клас, който представя една клетка от таблицата.

\* Всеки подтип трябва да реализира методите за показване и изчисление.

\*/

public abstract class Cell {

/\*\*

\* Връща текстовото представяне на клетката (за показване в таблицата).

\* @return стойност като текст

\*/

public abstract String getDisplay();

}

* IntegerCell

package core;

/\*\*

\* Клетка, съдържаща цяло число.

\*/

public class IntegerCell extends Cell {

private int value;

public IntegerCell(int value) {

this.value = value;

}

/\*\*

\* Връща стойността на клетката като текст.

\*/

@Override

public String getDisplay() {

return Integer.toString(value);

}

/\*\*

\* Връща стойността на клетката като число (double).

\*/

@Override

public double getValue() {

return value;

}

}

* FormulaCell

/\*\*

\* Класът FormulaCell представлява клетка, която съдържа формула.

\* Формулата може да бъде проста аритметична операция между две стойности

\* (числа или препратки към други клетки) или една стойност.

\*/

public class FormulaCell extends Cell {

private String formula;

private Spreadsheet spreadsheet;

/\*\*

\* Конструктор на клетка с формула.

\*

\* @param formula формулата, която тази клетка ще изчислява (напр. "=R1C1 + R2C2")

\* @param spreadsheet препратка към електронната таблица, нужна за извличане на други клетки

\*/

public FormulaCell(String formula, Spreadsheet spreadsheet) {

this.formula = formula;

this.spreadsheet = spreadsheet;

}

/\*\*

\* Връща стойността на клетката във вид, подходящ за показване.

\* Ако има грешка в изчислението, връща "ERROR".

\* Ако стойността е цяло число, връща без десетична точка.

\*

\* @return низова стойност за показване

\*/

@Override

public String getDisplay() {

double result = getValue();

if (Double.isNaN(result)) return "ERROR";

if (result == (long) result) return "" + (long) result;

return "" + result;

}

**4.2. Алгоритми и оптимизации**

*Алгоритъм за изчисление на формули:*

* Парсване на формулата
* Разпознаване на оператори и операнди
* Изчисление на стойностите
* Обработка на грешки

*Оптимизации*:

* Кеширане на резултати от формули
* Ефективно съхранение на данни
* Бързо изчисление на изрази

**5. Тестване**

**5.1. Планиране, описание и създаване на тестови сценарии (създаване на примери за всички възможни входове на командите)**

1. *Отваряне на файл*

* *open <локация на файла>*

1. *Преглед на таблицата*

* *print*

1. *Редактиране на таблицата*

* *edit 1 2 “Hello world!” - по този начин редактираме втора клетка от първи ред, като задаваме стойност Hello world! от тип string. Ако искаме да пишем число, пишем го без кавички, например: edit 1 2 45 - така втората клетка от първи ред ще е със стойност 45.*

1. *Формули*

* *за да приложим математически операции като събиране, изваждане, умножение, делене, пак използваме командата edit:*

*edit 2 4 = 10 + 12 - по този начин в четвърта клетка от втори ред ще се запише 24. Също можем да вършим математически операции между стойностите в две клетки, като напишем адреса на клетките. Например: edit 1 5 = R1C1 + R1C3*

1. *Запис на промените*

* *Ако искаме да запишем и запазим новите промени по таблицата, просто на конзолата се пише save.*

1. *Запиши като*

* *saveas <ново име> - по този начин се записва нов файл с въведеното от нас име.*

1. *Затваряне на таблицата*

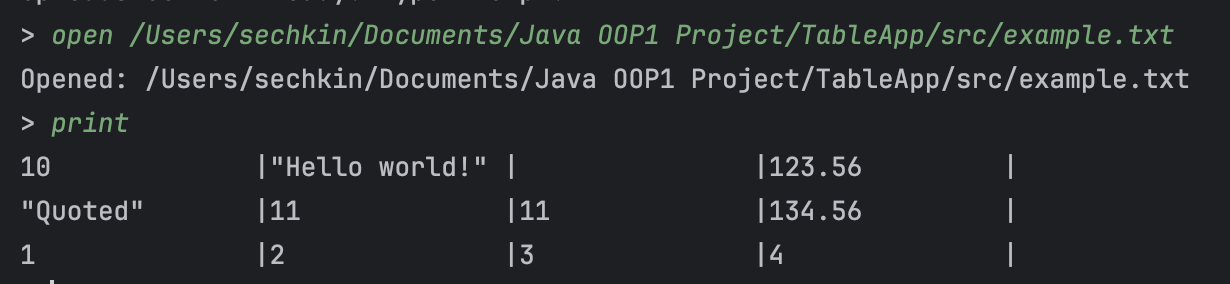
* *close*

1. *Изход от програмата*

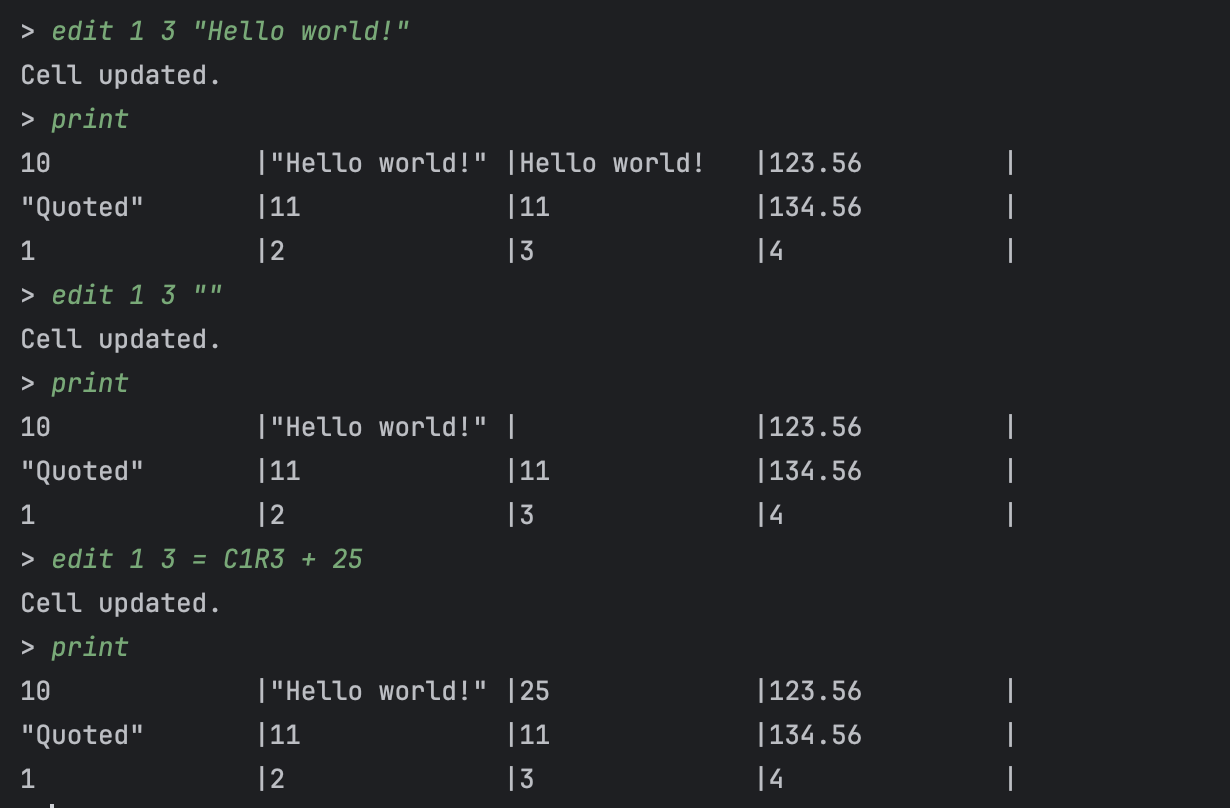
* *exit*

1. *Преглед на всички команди*

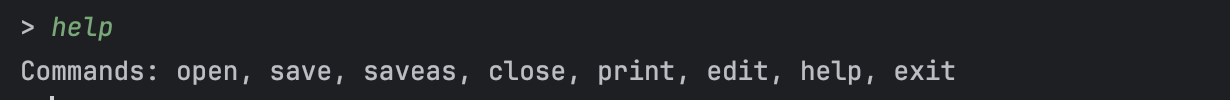
* *help*

***Примерни тествания:  
- отваряне на файл и принтиране на данните:  
***

* ***редактиране на клетка (Добавяне на стойности в клетка - текст, изтриване на стойността, прилагане на математически формули)***

******

* ***командата help***



**6. Заключение**

**6.1. Обобщение на изпълнението на началните цели**

Проектът успешно реализира всички поставени цели:

* Създадена е функционална електронна таблица
* Имплементирани са всички основни операции
* Добавена е поддръжка за формули
* Реализирана е файлова функционалност

***Използвана литература***

Java Documentation. Oracle. [Online] https://docs.oracle.com/en/java/

Object-Oriented Programming in Java. [Online] https://www.oracle.com/java/technologies/

CSV File Format. [Online] https://en.wikipedia.org/wiki/Comma-separated\_values