

**Технически университет – Варна**

Факултет по изчислителна техника и автоматизация

Катедра: „Софтуерни и интернет технологии”

Специалност: „Софтуерни и интернет технологии”

Тема: „Приложение за работа с електронни таблици”

Студент: Сезгин Бейханов Шабанов

Факултетен номер: 20621521

## Увод

**Основна идея:** Основната идея на приложението е да позволи на потребителя работа с електронни таблици. Като основни операции, които може да бъдат извършени са:

- Зареждане на вече съществуваща таблица или създаване на нова такава.
- Принтиране на таблицата на конзолата.
- Редактиране на таблица.
- Работа с формули.
- Записване на направените промени обратно в същия файл.
- Записване на направените промени в нов файл.

**Цел и задачи на разработката:** Основната цел беше да се направи напълно функционално приложение, което да позволява на потребителите лесна и ефективна работа с електронни таблици.

Задачите, които бяха разработени са:

- Прочитане на данните от текстов файл и вкарването им в електронна таблица.
- Валидация на типа данни, с които беше указано в заданието на проекта, че таблицата трябва да работи.
- Меню чрез, което се дава информация на потребителя как и какви операции може да извършва.
- Имплементация на функционалността **print**: за принтиране на таблицата на конзолата.
- Имплементация на функционалността **save**: за запазване на направените промени в текущо отворения файл.
- Имплементация на функционалността **save as**: за запазване на направените промени в нов файл.
- Имплементация на функционалността **edit**: за редактиране на таблицата.
- Имплементация на функционалност, която позволява на потребителя да работи с формули.

**Структура на документацията:** Структурата на документацията е следната. Заглавната страница включва информация за студента, факултета и темата на проекта.

**Увод** – тук се описват основната идея, цели и задачи на проекта.

**Преглед на предметната област** – описват се основни дефиниции, концепции и алгоритми, както и подходи и методи на решаване на поставените проблеми.

**Проектиране** – обща структура на проекта, диаграми/блок-схеми.

**Реализация и тестване** – реализация на класове, алгоритми и оптимизации. Създаване на тестови сценарий.

**Заклучение** – Обобщение, насоки за бъдещо развитие и усъвършенстване.

## Преглед на предметната област

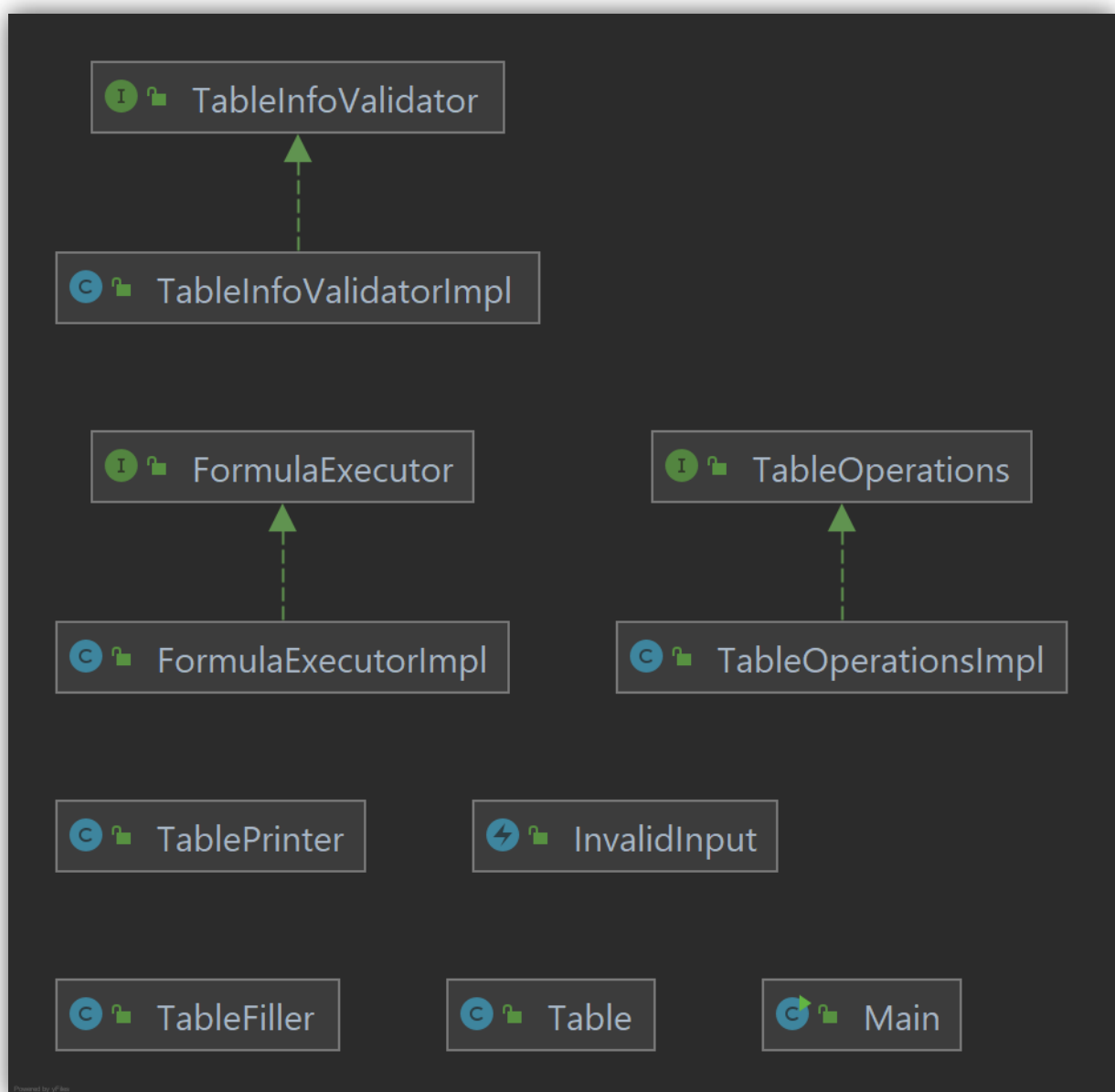
По време на разработка на проекта е обърнато внимание на това да се използват принципите на Обектно-Ориентираното Програмиране. В проекта за използвани принципи като: абстракция, полиморфизъм както и *SOLID* принципите. Обърнато е внимание на това проектът да е затворен за модификация и отворен за разширения, класовете и методите, са написани така, че да имат една единствена отговорност. Също така и на това всеки обект да може да се замени с бащиния тип без това да променя верността на програмата, както и на това интерфейсите да са по-малки по обем но повече на брой. И на последно място е приложен и принципът за това, че абстракциите не зависят на нищо, а конкретните класове зависят на абстракциите. Един от проблемите, които възникнаха по време на разработка беше валидацията на входните данни. За решаването на този проблем беше използван *regex*, за да може входните данни да са валидни така както е изискано да бъдат по условие. Друг проблем беше това, че входните данни не винаги идват по най-лесният начин за обработка. Този проблем беше решен като се правят няколко различни проверки, какъв е бил входа и според указаното в заданието как трябва да бъдат обработени данните, те биват обработени и представени в таблицата.

## Проектиране

Проектът се състои от три пакета

1. **src** – тук се намират класовете и интерфейсите както и другите два пакета от които зависи проекта.
2. **exceptions** – пакет за грешки.
3. **files** – пакет в който се намират текстовите файлове с примерни входни данни.

### Диаграма на класовете и интерфейсите в проекта



## Реализация и тестване

**Main** – Стартова точка на програмата. В този клас се обработват командите, които потребителя въвежда, за да избере каква операция иска да извърши. След като бъде въведена някоя команда **Main** делегира самата бизнес логика на класа, който е отговорен за свършването ѝ. Тук е и информацията за това какви операции поддържа приложението (менюто).

**Table** - Това е класът, който използваме, за да репрезентираме електронна таблица в приложението. Той има две частни полета, като първото поле носи информация за това, кой е файлът от който е заредена информацията за таблицата, а второто е матрица от низове, която репрезентира редовете, клетките и данните в тези клетки в таблицата.

**TablePrinter** – Това е класът, който има за отговорност да принтира таблицата на конзолата по възможно най *user friendly* начин.

**TableInfoValidatorImpl** – имплементира интерфейса **TableInfoValidator**, който има един единствен метод, който трябва да валидира данните, които ще бъдат попълнени в таблицата. Това валидиране на данните го постигаме, като използваме *regex* и няколко проверки, за това дали данните са: низ, цяло число, дробно число или пък формула. В случай, че данните са валидни те биват вкарани в таблицата, а в противен случай бива хвърлена грешката **InvalidInput**, която описва на кой ред в кой файл се е получила тази грешка и защо данната е невалидна.

**TableFiller** – Този клас го използваме, за да попълним таблицата с данни, които идват от файл. Също така този клас е зависим от **TableInfoValidator**, който е описан по-горе за какво се използва. Описание на някои от методите на класа **TableFiller**:

- **fillTableFromFile(File file, Table table)** - Този метод се използва, за да се прочетат данните, които се намират в даден файл и да се попълнят в таблицата.
- **arrangeTable(Table table)** – В условието не проекта е казано, че може да имаме изцяло празни редове. Този метод се използва, за да направи таблицата NxN матрица. Тоест колкото на брой са клетките с данни в най-запълнения ред, с толкова на брой клетки ще са всички редове в таблицата.
- **maxCellCountInTable(Table table)** – Методът връща цяло число, което репрезентира реда с най-много запълнени клетки в таблицата. Този метод се използва в други методи на класа, които са отговорни за това таблицата винаги да е подравнена, например в метода **arrangeTable(Table table)**.

**FormulaExecutorImpl** – имплементира интерфейса **FormulaExecutor**. Този клас има за отговорност да обработва формули. Като валидни формули за приложението се считат следните видове изрази:

- Две числа с аритметичен знак по средата.
- Число, аритметичен знак и адрес на клетка от таблицата
- Адрес на клетка от таблицата, аритметичен знак и пак адрес на клетка от таблицата.

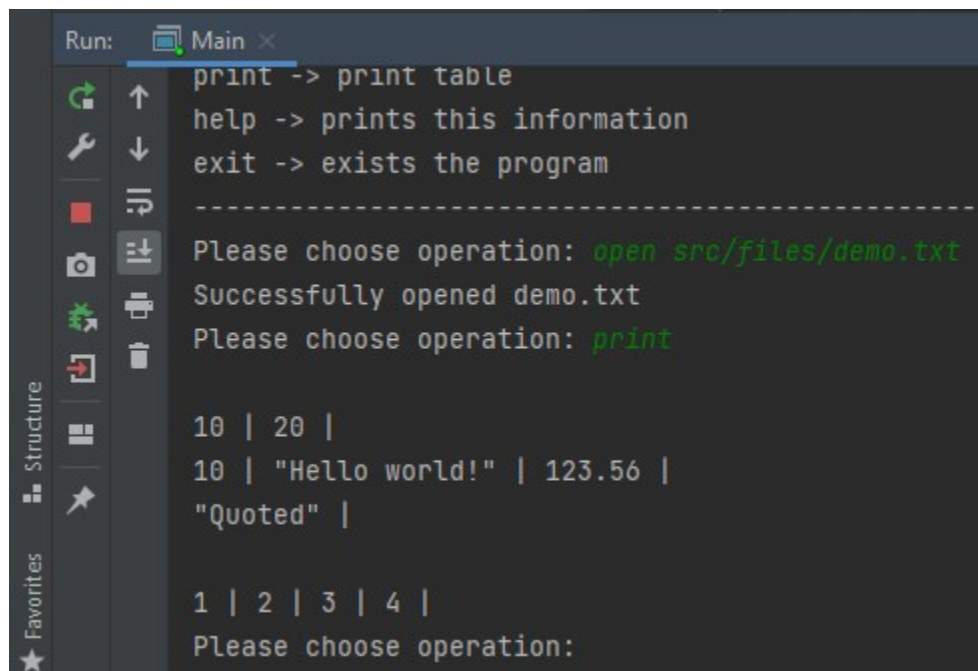
Като адреси на някоя клетка от таблицата се задават по следния начин: R1C1 – Първа ред първа клетка.

Ако съдържанието на дадена клетка е низ то стойността на клетката за формулата се конвертира до 0. Ако потребителя се опита да раздели някое число на 0 ще получи съобщение за грешка.

**TableOperationsImpl** – имплементира интерфейса **TableOperations** в който са дефинирани следните методи:

- **open(String filePath, Table table)** – Този метод е отговорен за това да отвори файла, от зададения път за четене. Ако такъв файл не съществува бива хвърлена грешка. В противен случай таблицата бива попълнена с данните, които се намират в този файл, като тази операция се делегира на метода **fillTableFromFile**, който се намира в **TableFiller**.
- **close()** - Този метод се използва, за да се затвори файла, който по-рано сме отворили.
- **save(Table table)** – Методът се използва, за да се запазят промените, които са направени в таблицата, в същия файл от който сме прочели данните.
- **saveAs(String filePath, Table table)** – Методът се използва, за да се запазят направените промени в таблицата, но не в същия файл от който сме прочели данните а в нов такъв. Това става чрез задаване на пътя на файла в който искаме да запишем промените.
- **edit(tableRowFromClient, tableColFromClient, String newDataForCell, Table table)** – Този метод получава като аргументи номера на реда и клетката, както и новата стойност, който искаме да запишем в тази клетка. Ако е подадено число, което надвишава броя на редовете в таблицата бива хвърлена грешка. В противен случай старата стойност на клетката бива изтрита и бива добавена новата стойност, която е подал потребителя.

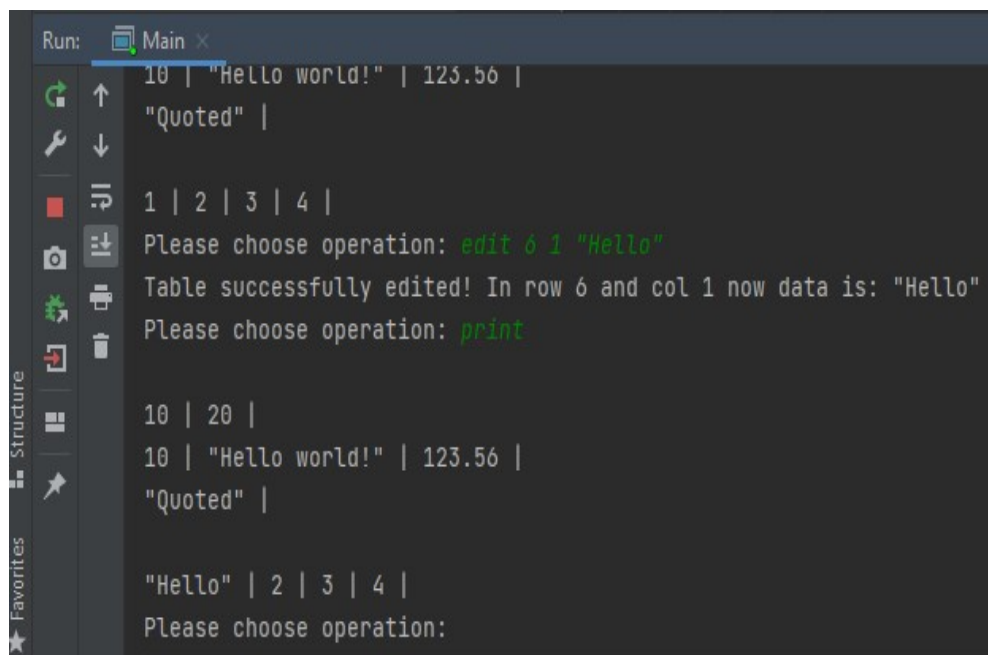
## Тестови сценарий за използване на приложението



```
Run: Main x
print -> print table
help -> prints this information
exit -> exists the program
-----
Please choose operation: open src/files/demo.txt
Successfully opened demo.txt
Please choose operation: print

10 | 20 |
10 | "Hello world!" | 123.56 |
"Quoted" |

1 | 2 | 3 | 4 |
Please choose operation:
```



```
Run: Main x
10 | "Hello world!" | 123.56 |
"Quoted" |

1 | 2 | 3 | 4 |
Please choose operation: edit 6 1 "Hello"
Table successfully edited! In row 6 and col 1 now data is: "Hello"
Please choose operation: print

10 | 20 |
10 | "Hello world!" | 123.56 |
"Quoted" |

"Hello" | 2 | 3 | 4 |
Please choose operation:
```

## **Заклучение**

Основната цел в началото беше да се направи приложение, което да улеснява работата на потребителите с електронни таблици. Основните функционалности, които бяха зададени в условието на проекта бяха успешно имплементирани. За бъдещето развитие на приложението е възможно да се добавят и други видове файлове от които да се чете информацията за таблицата, защото в момента то е ограничено само до файлове, които са от тип txt. Както и например добавяне на нови функционалности към приложението като: добавяне на нова стойност във вече съществуваща таблица, намиране на средноаритметичната сума на данните в таблицата, например намиране на средноаритметичната сума на всички данни от първи ред.