Технически Университет Варна

Документация

Курсов Проект ООП – Част 1

тема: **Star Wars Universe 0.1**

Изготвил: Ниляй Рамадан

фак. номер: 21621575

курс: 2

група: 1 а)

Съдържание.

I. Увод

1. Описание и идея на проекта

Проектът „Star Wars Universe 0.1“ има за цел да създаде една малка система от планети и Джедаи, които ги населяват. Потребителят може да извършва операции върху отвореният *XML* файл, като например да добави планета или Джедай, както и да изтрива Джедаи.

След стартирането на програмата, потребителят трябва да посочи коя команда иска да се изпълни, като съответно първо е нужно да има отворен файл, с който да се работи. Щом файлът се отвори, данните, които са запаметени в него се зареждат, а всяка една нова промяна се запаметява само в паметта, освен ако потребителят не пожелае да запази данните в отворения или в някой друг файл, като изрично се посочва абсолютният път, на който се намира файлът.

Друга изпълнима команда е затваряне на отворения файл, без да се запишат някакви промени.

В случаите, в които потребителят въвежда грешна команда, която я няма в менюто, то на екрана се изписва съобщение-подсказка, която му посочва, че може да намери всички команди в ‘*help’* опцията. Това подменю изписва на екрана всички налични команди, от които той може да избере и изпълни. Срещу всяка една от тях се изписва и за какво тя служи и какъв трябва да бъде синтаксисът.

2. Цел и задачи на проекта.

Целта на проекта „Star Wars Universe 0.1“ е да предостави на потребителя възможността да изпълни функционалности като добавяне на нова планета, да създаде нов Джедай, който да населява една от тези планети, да запамети/запише неговите характеристики.

Основните команди, които са извън контекста на темата на проекта са отваряне, затваряне, записване и помощ. Освен тези, проектът трябва да съдържа и дава възможност на потребителя да избира от командите, които са пряко свързани с темата. Това са командите за:

- добавяне на нова планета
- създаване на нов Джедай
- премахване на вече съществуващ Джедай
- повишаване на ранга на Джедай
- понижаване на ранга на Джедай
- извеждане на най-силния Джедай
- извеждане на най-младия Джедай
- извеждане на най-често срещаният цвят на светлинния лъч на Джедая
- извеждане на най-често срещаният цвят на светлинния лъч на Джедая, който се ползва от поне един ‘*GRAND MASTER’*
- извеждане на дадена планета и населяващите я Джедаи
- извеждане на информация за съответния Джедай, както и коя планета населява
- извеждане на всички Джедаи, населяващи две специфични планети, избрани и въведени от потребителя

Всяко едно от свойствата както на планетите, така и на Джедаите има създадени методи за верификация, за да може въведените от потребителя данни да бъдат коректни и валидни за по-нататъшно изпълнение. Целта на тези методи за верификация е програмата да продължава да поиска нужната информация повторно, докато тя не стане валидна според изискванията за съответното свойство. По този начин програмата гарантира изцяло валидни данни, които могат да се записват и прочитат от XML файлове.

При изпълнението на всяка една от горепосочените опции, ако се въведе невалидна команда или команда, която не съществува на потребителя му се извежда нужното съобщение, но приложението не прекратява своето изпълнение, докато не се избере командата за Изход – '*Exit'*.

3. Структура на документацията.

Документацията е структурирана в няколко глави и представя проекта като последователност от стъпките за създаването.

Първата глава, наречена „Увод" съдържа описанието и идеята на проекта, целта и задачите на разработка, както и структурата на документацията. В тази част на кратко се обяснява идеята на проекта, неговото кратко съдържание и възможностите, които проектът открива пред потребителя.

Втората глава, за по-кратко наречена „Преглед на предметната област", представя основни дефиниции, концепции и алгоритми, които са използвани, както и дефиниране на проблеми и сложност на поставената задача. Следователно в тази част от Документацията, следва да се опишат и начините и подходите за решаване на срещаните проблеми по време на изпълнението на задачата.

В следващият дял от Документацията се разглежда Проектирането на проекта. Съответно тук се описват пакетите, които се ще реализират, диаграмите и блок схемите, които описват структурата и поведението на обектите.

В четвърта глава, наречена „Реализация и тестване" се включват реализацията на класовете, както и важни моменти и кодови фрагменти. Друга важна точка тук е и планирането и създаването на тестовите сценарии, с цел показване на функционалността на програмата в различни ситуации, които могат да се създадат по време на използването на програмата от различните потребители.

В заключителната последна глава се включва обобщение на изпълнението на началните цели, както и насоки за бъдещо развитие и усъвършенстване на проекта.

II. Преглед на предметната област

1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани.

Команда. (Command)

Интерфейсът „Команда“ декларира метод ‘runCommand’, който е предназначен за изпълнението на дадена команда. Този интерфейс се имплементира в главното меню, където се извикват нужните методи за изпълнение на съответната операция, избрана от потребителя.

Методът ‚runCommand‘ е от тип ‚void‘, тъй като той служи единствено като средство за извикване на дадена команда, а последващата изпълнение е от страна на този определен метод, който трябва да информира потребителя за по-нататъшното изпълнение.

Моята Система (MySystem)

Класът е част от създаването на система за обработка на команди, а именно Командният Интерфейс, където потребителят може да изпълни избраната от него опция и да получи съответния отговор.

Използва се дизайн шаблонът от групата на Създаващите шаблони, а именно този за една единствена инстанция от даден обект – *‘Singelton’*. По този начин се предоставя възможността за глобален достъп до системата чрез метода ‚*getInstance*‘. Причината за избор на точно този шаблон е удобният случай, когото трябва да се осигури, че в цялата система има само една точка на управление.

Чрез метода старт ‚*start‘* потребителят може да въвежда команди, които се четат от стандартния вход, а именно клавиатурата в този случай. Въведената команда се разделя на отделни части или така наречените аргументи чрез метода ‚split‘. Този подход позволява на потребителя да въвежда команди с параметри, които програмата ще разграничи и съответно раздели, за да може да се изпълни правилно избраната команда. А изходът от системата не настъпва, докато потребителят не въведе ‚exit‘.

Главно меню(MainMenu)

Класът имплементира интерфейса Команда и съответно неговият метод *runCommand',* който обработва различни команди, предоставени от потребителя чрез командния ред. Методът приема като параметър масив от низове, наименуван ‚*options'*. Първият елемент от този масив, съответно този с индекс 0, е командата, която трябва да се изпълни. Това е и причината в условния оператор за много избори като селектор да се използва именно първият елемент.

Тук се използва условният оператор за взаимоизключващи се възможности, '*switch-case*', който предоставя възможност за избиране на правилната команда, която е въведена от потребителя и съответно изключване чрез оператора ‚break', след края на изпълнение на дадената команда. В случаят по подразбиране, т.нар. ‚*default*', се попада единствено, когато потребителят въведе команда, която не съществува или я изписва грешно. За да се намали броят на случаите, в които попадаме в този сценарий, се използва методът ‚*toLowerCase*', за да може да не ограничаваме потребителя и съответно не създадем излишни грешки в работата на програмата.

Създадената булева променлива дали файлът е отворен или не ‚isFileOpened' служи на програмата, за да осигури файл, който да е отворен, тъй като всяка една команда трябва да може да се изпълнява единствено, ако вече има зареден файл. В случаите, в които потребителят избере различна команда, а все още няма отворен файл, то тогава на екрана ще се изпише следното съобщение, което ще го информира: „Няма отворени файлове! Моля, изберете опцията „помощ", за да видите всички налични команди.".

Когато потребителят избере команда за затваряне на текущия отворен файл, но такъв не съществува, то тогава тази грешка се „хваща" и на екрана отново се изписва нужната информация, която информира потребителя, че трябва първо да използва командата за отваряне, за да може да избере и последващата команда за затваряне.

Командата за изход от програмата е ‚exit ', която съответно позволява на потребителя да излезе от системата, в случаите, в които не желае да продължи да използва приложението.

2. Дефиниране на проблеми и сложност на поставената задача

Проблем 1. Четене и запис в XML файл.

Работата с XML файловете беше малко по-различна от използваните до сега файлове и за това възникна въпросът как точно трябва да се работи с тях, за да може да няма грешки при четенето и съответно записването на данните.

Използването на правилните етикети (тагове) е една от най-важните стъпки, за да не спре работата на приложението, когато се работи с такъв тип файлове.

Проблем 2. Валидиране на данните, въведени от потребителя.

Създаването на планети или Джедаи води до проблема за валидиране на входните данни, които трябва да въведе потребителя от стандартния вход.

- необходимо е валидиране на името на планетите и Джедаите, за да може в последствие да не възникнат грешки при търсенето и намирането им, тъй като последващите методи изискват работа със специфични обекти и единственото им уникално свойство е името.

Проблем 3. Повишаване и намаляване на ранга на Джедай

Джедаят има свойството ранг, които са точно определени и не могат да бъдат избрани и въведени произволно от потребителя.

- Съответно тук може да възникне проблемът за грешно въведен ранг, който при прочитане на файла не може да се открие като стойност в ‚enum‘ класа за ранг на Джедаите.
- В този случай би било трудно и повишаването или намаляването на ранга на даден Джедай.

Проблем 4. Изпълнението на командата за извеждане на всички Джедаи, населяващи двете, избрани от потребителя планети.

Въвеждането на планети, чиито имена на са ясни по време на програмирането на приложението, отвори въпроса за това как точно ще се гарантира, че менюто ще съдържа тази команда.

Тъй като в Главното меню списъкът от планети е все още неясен, а и може да се работи с файл, в който да няма никакви данни за извличане, то тогава трябва да се открие начин, за да не се създават проблеми и грешки при изпълнението.

3. Подходи и методи за решаване на поставените проблеми.

Подход към решение на проблем 1. Четене и запис в XML файл.

Създаденият клас за четене на файлове – '*ReadXMLFiles*' създава обект Файл за зададеното като параметър име на файла. Използват се '*DocumentBuilderFactory*' и '*DocumentBuilder*' за създаването на XML файловете. Първоначално се извличат елементите с етикет ‚planet', като всяка една планета се добавя към списъка от планети, наречен ‚*listOfPlanets*'.

Следващата стъпка е прочитането и на всички Джедаи, населяващи съответните планети, заедно с техните свойства, като име, години, цвят на светлинния меч, ранг и сила. След успешното прочитане на съответния Джедай, то той се добавя към списъка от населяващите всяка планета Джедаи.

Ако не се намери нито една планета, то тогава се извежда съобщение, че файлът е отворен успешно, но в него няма намерени данни за извличане, тъй като няма как да има съществуващи Джедаи, ако няма и планети, които те да населяват.

Подход към решение на проблем 2. Валидиране на данните, въведени от потребителя.

Решението, което намерих е създаването на отделни методи за проверка на входа. Следователно всяка едно от свойствата на Джедая, както и за името на планетата има отделен метод, който съдържа 'do-while' цикъл. Той гарантира, че при всеки един невалиден вход от потребителя, ще се покаже необходимото съобщение и системата ще изчака за нов вход от клавиатурата.

- за да се гарантира уникалност на имената, е нужно да се създаде метод, който да провери дали вече има такова съществуващо име и в случай, че има нека се изведе подходящото съобщение

По този начин се осигурява правилно въведена информация, която в последствие може да се запише във файл или да се чете при отваряне на файла.

Подход към решение на проблем 3. Повишаване и намаляване на ранга на Джедай

Клас от тип ‚*enum*' се използва за по-лесен достъп и използване на ранговете, от които може да притежава един Джедай. Това улеснява избора на потребителя и също така намалява възможността за срещане на грешки при добавяне на нов Джедай.

- За да не се въведе с правописна грешка рангът, което от своя страна ще доведе до грешка при прочитане на файла, е създадена структура от тип 'Map', която се използва за съпоставяне на целочислени стойности със съответните стойности от ‚*enum*'-а, който съдържа наименованията на всички рангове.

Всеки целочислен ключ представлява конкретно ниво(1-8), а асоциираната стойност е съответстващата му от изброимия тип клас. По този начин, от потребителя се изисква единствено въвеждането на номера на ранга, от който иска да бъде неговият Джедай, а в случаите, в които се въвежда число по-малко от 1 или по-голямо от 8, то от потребителя се очаква повторно въвеждане на правилния вход.

- За повишаването или намаляването на Джедая се изчислява новият ранг чрез използването на индекса на позицията(‘ordinal’) на дадения елемент от класа, а след това се добавя единица, за да може реално да се увеличи рангът. Използва се готовият метод за минимум/максимум, за да се подсигури, че ако вече е най-високият/ниският ранг, да се изпише необходимото съобщение, а силата да не се променя според дадената формула.

Подход към решение на проблем 4. Изпълнението на командата за извеждане на всички Джедаи, населяващи двете, избрани от потребителя планети.

При изграждането на структурата на главното меню всички команди освен последната са ясно наименувани и имат очакван вход, докато при последната нещата не стоят точно по този начин. Все още не се знае кои ще бъдат планетите, които ще създаде потребителят, нито с кой файл ще се работи. Затова единственото правилно решение, което позволява на програмата да работи правилно е създаването на отделна условна проверка, която да провери дали вторият вход с индекс 1 е равен на знака ‚+‘. Единствено в тези случаи това условие ще бъде изпълнено и ще изведе всички Джедаи, населяващи двете планети в сортиран лексикографски вид.

А за да не влиза и в многоселекторния оператор, попадайки в случая за подразбиране и изписвайки съобщение, че подобна команда не съществува, се използва флаг, който първоначално се декларира като лъжа, а при влизане вътре в тялото на условието, то се променя на истина.

По този начин всяка една от командите може да се изпълни от програмата, дори и без да има достъп до списъка от планети.

4. Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...) и качествени (нефункционални) изисквания (скалируемост, поддръжка, ...)

Функционални изисквания

Приложението трябва да позволи на потребителя да отваря файлове и да извлича данни от тях, като съответно да може и да работи с тези данни, като например да намери най-младия Джедай или да повиши ранга на даден Джедай, както и да промени силата му по дадена формула.

Потребителят трябва да има възможността да затвори файла без да запише направените промени или да запише промените в текущия или друг файл. Ако потребителят изрично не каже „запази“ или „запази като“, то след затваряне на файла няма да има никакви промени във файла.

Интерфейсът на командния ред позволява лесен достъп до всички команди, дори и когато възникне грешка, то ще се изведе съобщение, което да насочи потребителя, за да не приключи той работата с приложението.

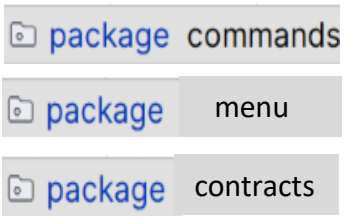
Нефункционални изисквания

Потребителски интерфейс: Приложението трябва визуално да изглежда добре и съответно да позволява лесно прочитане на изведената информация за потребителя.

Обработка на грешки: При появата на неочаквани грешки, които много вероятно могат да възникнат, тъй като се работи с външни файлове, чието съдържание не е ясно, то трябва да се изведе необходимото съобщение на екрана без да се прекъсне работата на програмата.

III. Проектиране

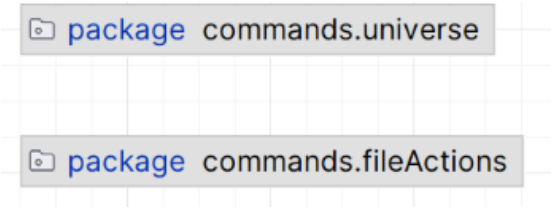
1. Обща структура на проекта. Пакети, които ще се реализират.



Извън тези пакети се намира и главният клас, който играе ролята на главен клас в приложението – *'Main'*. Той съдържа метода '*main*', който е входната точка при стартиране на програмата.

Пакетът *'commands*' включва в себе си два подпакета – '*fileActions'*, *'universe'*.

Първият, от които е отговорен за командите, нужни при работа с файлове, като отваряне, затваряне, запазване. Докато вторият, наречен за по-кратко „Вселена" е отговорен за командите, свързани с планетите и населяващите ги Джедаи.

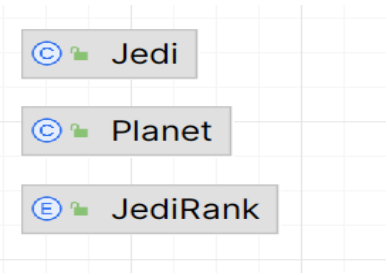


Пакетът '*universe* ' съдържа в себе си най-важният клас за цялото приложение, а именно '*Jedi*', както и '*Planet*' и класът от тип '*enum*' - '*JediRank*'.

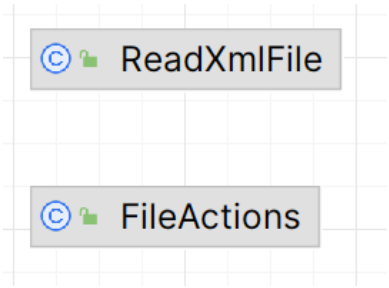
Класът '*Jedi*' е отговорен за създаването и валидацията на свойствата на Джедаите. Тук също така са и методите, които са пряко свързани с Джедаите и командите, които може да въведе потребителят.

Класът '*Planet*' е отговорен за създаването на планетите, както и тяхната валидация, което както при Джедаите и тук включва уникалността на всеки един обект.

Класът от тип *enum*' - '*JediRank* съдържа в себе си всички рангове, които може да притежава един Джедай, както и двата метода, свързани с повишаването и намаляването на ранга на съответния Джедай.



Пакетът ‘*fileActions’,* включва всички необходими команди, които са нужни при работа с файлове. В себе си той съдържа два класа – първият, съдържа в себе си методите за отваряне, затваряне, записване или помощ. Докато вторият е този, който се извиква в метода за отваряне на файлове, а именно за да прочита и извлича данните, ако има такива вътре във файла.



Пакетът *‘menu’* съдържа в себе си два класа, които са необходими при стартирането на програмата, тъй като са пряко свързани с главният метод, а именно ‘*main*’.

Първият клас наречен ‘MySystem’, допринася за използването на потребителския интерфейс на командния ред, чрез който потребителят може да въвежда командите, които иска да изпълни програмата.

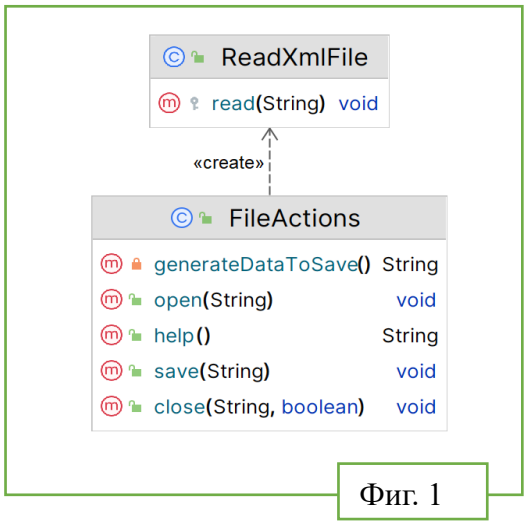
Вторият клас, наименуван ‘MainMenu’, е главното меню, в което се съдържат всички налични операции, които могат да бъдат изпълнени и съответно извиква специфичния метод.

Пакетът ‘contracts’ съдържа в себе си двата интерфейса, нужни на приложението, а именно за изпълнение на командите – ‘Command’ и за добавяне на Джедай – ‘Add’.

2. Диаграми/Блок схеми (на структура и поведение - по обекти, слоеве с най-важните извадки от кода).

На Фиг. 1 е показана блоковата схема на класовете, използвани при работа с XML файлове. В класа *'ReadXMLFile'* се съдържа методът за прочитане на файла, който се извиква директно в метода за отваряне на файл – *'open'*.

Методът *'generateDataToSave'* се използва, за да може данните, които се записват в XML файла да притежават правилните етикети, а след това и да могат да се прочитат правилно при отваряне на даден файл. Също така тук се запълва и списъкът на планетите с населяващите ги Джедаи, ако има такива.

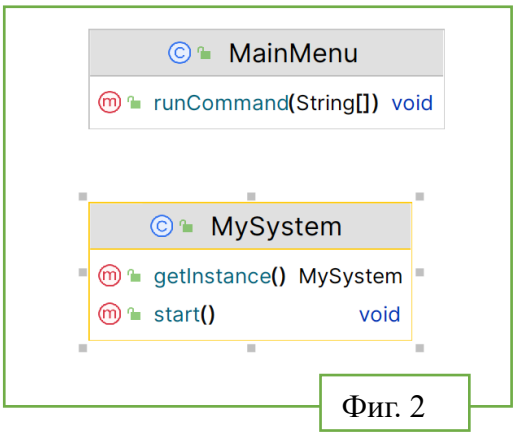


Фиг. 1

На следващата фигура №:2, е показано менюто, което служи за стартиране на програмата, както и стартиращият метод – *'Main'*.

Класът *'MainMenu'* е онзи клас, който разпределя и открива командата, която потребителят е въвел и се грижи за безупречното изпълнение на командите, дори и когато възникнат грешки, за да може потребителят да продължава да работи със системата без прекъсвания.

В класа *'MySystem'* се използва създаващият шаблон за дизайн – '*Singelton*', който осигурява една единствена инстанция на стартиращата система.



Фиг. 2

В пакета „Вселена“ се намират двата най-съществени класа - ‘Jedi’ и ‘Planet’, в които се намират всички онези методи за получаване на свойствата им, както и за инициализиране на стойностите на тези свойства.

Фиг. 3

IV. Реализация, тестване.

1. Реализация на класовете. Алгоритми.

Първостепенният клас, от който трябва да се започне обяснението на кода, смятам, че е *'MySystem'*, тъй като именно той е основополагащият за потребителския интерфейс на командния ред. Тук се създава статична инстанция на класа, като се използва създаващият шаблон за дизайн, който ще осигури една единствена такава система.(Фиг. 4)

Методът 'start' подпомага за работата на командите, които потребителят въвежда, защото дори и когато има грешки програмата няма да преустанови работа, а само ще изведе съобщение за съответната грешка, позволявайки на потребителя да опита отново без да загуби направените промени. Това се постига чрез използването на блок за прихващане и обработка на изключения/грешки – *'try-catch'*. (Фиг.5)

Докато потребителят не въведе команда за изход от програмата, а именно 'exit', то изпълнението на приложението ще продължава, запазвайки всички промени поне в хранилището, ако команда за запазване не е била изрично указана.

```
    public void start()  {
        Command command = new MainMenu();
        Scanner scanner = new Scanner(System.in);
        String input;
        String[] option;
        while(true){
            System.out.println("> ");
            input = scanner.nextLine();
            if(input.equals("exit")){
                System.out.println("Exiting the system!");
                System.exit(1);
            }
            option = input.split(" ");
            try{
                command.runCommand(option);
            }catch (Exception e){
                System.out.println(e.getLocalizedMessage());
            }
        }
    }
```

Фиг. 4

```
public static MySystem getInstance(){
   if(instance == null){
         instance =  new MySystem();
     }
    return instance;
}
```

Фиг. 5

Следващият важен момент в разглеждането на кода е създаването на главното меню. Всяка една от командите, за работа с файлове, ще бъде разгледана малко по-надолу в документацията, докато тук е време да спомена структурата на менюто. Използва се многоселекторният условен оператор '*switch'*, който при несъвпадение с нито едно от поставените условия попада в избора по подразбиране, който съобщава на потребителя, че подобна команда не съществува и го съветва да погледне подменюто за помощ.

Както беше показано и на Фиг. 4 масивът от низове наименуван '*option*' се подава като параметър на метода за изпълнение на командата ‚*runCommand*', а след това и като селектор на условния оператор.

```
switch (option[0].toLowerCase()) {…
```

Започвайки с най-важната команда, използвана при работа с XML файлове, трябва да спомена, че отварянето на файлове трябва да съдържа и проверка дали такъв файл съществува или не. В случаите, в които такъв не е открит, то тогава той трябва да бъде създаден. Ако се открие такъв файл, то неговото съдържание трябва да бъде извлечено и да се покаже подходящото съобщение на потребителя.

Използва се отново блокът за прихващане и обработка на грешки/ изключения– *'try-catch'*. Той подпомага за правилното изпълнение на метода, дори и когато се открие грешка, то програмата не спира своето изпълнение, а на потребителя му се изписва подходящо съобщение.

```
try {
      if (!Files.exists(Path.of(filePath))) {
            Files.createFile(Path.of(filePath));
            System.out.println("File " + fileName + " created
                  successfully.");
      } else {
      reading.read(filePath);
           System.out.println("File " + fileName + " opened
                successfully.");
         }
  } catch (IOException e) { e.printStackTrace();
  }
```

Затварянето на файл означава прекратяване на работа с него и докато не се отвори текущият или друг файл не може да се изпълни друга команда освен „помощ“. В случаите, в които няма отворен файл, то на екрана се изписва подходящо съобщение, което да информира потребителя. След успешно затваряне списъкът с планети и населяващите ги Джедаи се изпразва и на екрана се показва съобщение, че текущият файл е бил затворен, като се изписва и неговото име.

```
      if(!isFileOpened){
    throw new Exception("No files opened! Please use 'open' command and
write the name of your file.");
}else{
    Planet.listOfPlanets.clear();
    System.out.println("Closed file: " +fileName);
}
```

Булевата променлива, която се използва почти във всяка една команда освен тази за отваряне и помощ, подсигурява, че докато няма отворен файл, то няма как да се изпълни дадената опция.

Следващата команда е тази за „помощ“, която изписва на екрана всички възможни опции, от които потребителят може да избере за изпълнение. Тук по най-простият начин се инициализира един низ, наречен резултат, който съдържа кратка информация за това за какво служи всяка една от командите.

```
String helpInfo = "These are all the commands you can run: \n" +
        "open (file) -> open a file after entering its name \n" +
        "close -> closes currently opened file \n" +
        "save -> saves the currently opened file \n" +
        "saveas -> saves the currently opened file in a directory or another
file \n" +
        "help -> prints commands information \n" +
        "add_planet -> add a new planet \n" +
        "create_jedi -> create a new Jedi \n" +
        "remove_jedi -> remove a jedi from a planet \n" +
        "promote_jedi -> let the jedi get better and promote his rank and
strength \n" +
        "demote_jedi -> demote the jedi a level down on the rank list and
lower the strength \n" +
        "get_strongest_jedi -> get the strongest jedi of all Jedis on the
planet \n" +
        "get_youngest_jedi -> get the youngest jedi on the planet with a
specified by you rank \n" +
        "get_most_used_saber_color -> get the most used saber color by jedis
on the planet and with a specified by you rank \n"  +
        "get_most_used_saber_color_grand_master -> get the most used saber
color used from at least one Grand Master \n" +
        "print_planet -> print jedis on a planet sorted by rank and then
lexicographically \n" +
        "print_jedi -> print jedi details and on which planet it is located
\n" +
        "planetName + planetName -> get the jedis on two planets sorted
lexicographically \n" +
        "exit -> exits the program";
return helpInfo;
```

Както е споменато и малко по-нагоре не всяка една промяна се запазва, освен ако тя не е изрично посочена от потребителя. Командата *'save'* използва пътя на текущия отворен файл, за да запише всички промени, които могат да бъдат създаване на планета, създаване на нов Джедай или изтриване на такъв. Тук се използва и помощният метод, който взима текущия списък с всички направени промени и го записва във файла чрез използването на метода за добавяне '*append*'.

```
for (Planet planet : Planet.listOfPlanets) {
    data.append("\t<planet>\n");
    data.append("\t\t<planetName>").append(planet.getName()).append
    ("</planetName>\n");
    data.append("\t\t<jedis>\n");
    for (Jedi jediOnPlanet : planet.getJedis()) {
        data.append("\t\t\t<jedi>\n");
        data.append("\t\t\t\t<jediName>").append(jediOnPlanet.
        getName()).append("</jediName>\n");
        data.append("\t\t\t\t<jediAge>").append(jediOnPlanet.
        getAge()).append("</jediAge>\n");
        data.append("\t\t\t\t<lightSaberColor>").append(jediOnPlanet.
        getLightsaber()).append("</lightSaberColor>\n");
        data.append("\t\t\t\t<rank>").append(jediOnPlanet.
        getRank()).append("</rank>\n");
        data.append("\t\t\t <strength>").append(df.format(jediOnPlanet.
        getStrength())).append("</strength>\n");
        data.append("\t\t\t</jedi>\n");
    }
    data.append("\t\t</jedis>\n");
    data.append("\t</planet>\n");
}
data.append("</planets>\n");
```

Другата команда за запазване е тази, която служи за запазване в друг файл с различен път или за презаписване на информацията в друг файл с текущата информация, съдържаща се в хранилището. Тук е нужно да се изпише целият път на файла, в който ще се запише информацията. За да не се повтаря отново кодът се използва отново методът за запазване ‘save’, като веднага след това се очаква от потребителя да напише абсолютният път към файла, в който иска да запише направените промени.

```
if (isFileOpened) {
   System.out.println("Please enter the path you would like to save your file: ");
   String filePath = scanner.nextLine();
   file.save(filePath);
} else {
   System.out.println("No files opened! Please check 'help' to see all commands”);
}
```

Следващата важна стъпка е разглеждането на командите свързани с темата на проекта, а именно Джедаите и планетите.

Първоначално трябва да обърнем внимание на това как в един списък, който съдържа имената на създадените планети можем да включим и всички населяващи ги Джедаи. Това се случва като се създаде един списък от тип ‘*ArrayList*’, наименуван ‘*listOfPlanets*’, а вътре в него се влага втори, който е от тип ‘*List*’ – ‘*jedis*’.

```
public static ArrayList<Planet> listOfPlanets = new ArrayList<>();
private List<Jedi> jedis;
```

Списъкът от планети се декларира като статичен, за да позволи достъп до него и извън инстанциите на класа, като се избегне необходимостта от създаване на инстанции на класа. Това прави списъка "глобален" в рамките на класа, което означава, че всички инстанции на класа и други класове могат да достъпват и манипулират този списък, без да е необходимо да създават инстанция на класа ‘Planet’. Това осигурява уникалност на списъка от планети и гарантира, че той ще бъде винаги актуален и уникален, т.е. единствен.

За да се гарантира уникалността и на планетите, то се използва създаденият метод - ‘isNameUnique’. Той обхождайки целия списък проверява дали името, въведено от потребителя се среща или не, а в случаите, в които това име вече е налично, на екрана се изписва подходящо съобщение, което ще му позволи да избере ново име, за да създаде планета.

Друг важен метод пък проверява дали името е валидно, т.е. дали името не е просто цифра, дали не е просто празен интервал, тъй като това няма да позволи правилната работа на метода за уникалност на имената. Този метод се използва и при проверка на имената на Джедаите, както и някои други свойства, които са низове.

```
private boolean isNameValid(String name) {
        return !name.isBlank() && !Character.isDigit(name.charAt(0));
}
```

Методът за проверка на името на планетата е всъщност онзи метод, който първи сработва, когато потребителят иска да добави нова планета, избирайки опцията '*add_planet'*. Тук се извикват и двата метода за проверка на уникалност и валидност на въведеното име, като се използва форма на рекурсията, за да не се наложи потребителят отново да въведе в командния ред опцията за добавяне на планета.

```
do {
    System.out.println("Please enter the name of your planet: ");
    name = scanner.nextLine();
    if (isNameValid(name)) {
        if (isNameUnique(name)) {
            result = name;
        } else {
            System.out.println("UUPS! A planet with this name already
exists!");
            result = checkName();
        }
    }
} while (!isNameValid(name));
```

След успешно добавяне на планета, на екрана се изписва, съобщението, че добавянето на съответната планета е било успешно.

Другата команда за добавяне е тази за Джедаите. Тук има доста повече свойства на героите, което от своя страна изисква по-голям брой проверки за валидация, за да може програмата да работи с правилни по тип данни.

Едно от най-важните и отново уникални неща е името на Джедая. Тук се прави проверка за уникалност, т.е. дали друг Джедай със същото име съществува и дали въобще въведеното име е валидно. Същите тези проверки се правят и при избора на цвета на светлинния меч на Джедая.

За възрастта на Джедая също се прави валидация, за да се избегне въвеждане на низ от символи или число по-малко от нула. Проверка за правилна стойност на силата на дадения Джедай се прави също, тук дори и да се въведе цяло число, то то ще бъде отново конвертирано към число с плаваща запетая, според изискванията на условието на проекта.

По-интересното валидиране е при избора на ранг. Тук, за да се гарантира, че единствено ще се въведат онези стойности, които са включени в класа от тип ‘*enum*’ – ‘*JediRank*’, се създава *Map* (Фиг.6), който когато дойде ред на избор на ранг се изписва на екрана по номериран ред и единственото, което се изисква от потребителя е въвеждането на съответния номер. Докато при принтиране на Джедая на екрана това число се конвертира към своята стойност в структурата ‘*Map*’ (Фиг.7).

Фиг. 6

```
public Map<Integer, JediRank> getRankList() {
Map<Integer, JediRank> possibleRanks = new HashMap<>();
    possibleRanks.put(1, JediRank.YOUNGLING);
    possibleRanks.put(2, JediRank.INITIATE);
    possibleRanks.put(3, JediRank.PADAWAN);
    possibleRanks.put(4, JediRank.KNIGHT_ASPIRANT);
    possibleRanks.put(5, JediRank.KNIGHT);
    possibleRanks.put(6, JediRank.MASTER);
    possibleRanks.put(7, JediRank.BATTLE_MASTER);
    possibleRanks.put(8, JediRank.GRAND_MASTER);
    return possibleRanks;
}
```

```
if (currentRank < 1 || currentRank > 8) {
    System.out.println("Please write down a number between 1-8
for your Jedi's           rank!");
} else {
    result = getRankList().get(currentRank); // Adjust index
    validRank = true;
}
```

Фиг. 7

Най-важната част е населяването на избраната планета. Тъй като има вероятност потребителят да не знае кои са текущите планети, то когато дойде ред на избор на планета на екрана се изписват всички планети отново чрез използването на структурата *Map* (Фиг.8) и потребителят трябва само да въведе номера на планетата. Тук обаче има и друга възможност за добавяне на нова планета без да се прекъсне добавянето на Джедая. Това се случва чрез избирането на числото „0“ , което автоматично извиква метода за добавяне на нова планета.

```
int counter = 1;
for (Planet planet : Planet.listOfPlanets) {
    planetNumberMap.put(counter, planet.getName());
    counter++;
}
for (Map.Entry<Integer, String> entry : planetNumberMap.entrySet()) {
    System.out.println(entry.getKey() + ". " + entry.getValue())
}
```

Фиг. 8

Тъй като в тази структура номерирането започва от единица, докато в листа от масиви, където са планетите, то започва с индекс 0, при избора на планета се изважда единица от получения резултат.

```
Planet selectedPlanet = Planet.listOfPlanets.get(number - 1);
selectedPlanet.addJedi(jedi);
```

По този начин се добавя създаденият Джедай към онази планета, която потребителят е посочил. След добавянето му на екрана отново се изписва подходящо съобщение.

Друга важна команда е за премахване на вече съществуващ Джедай – '*removeJedi*'. След въвеждането на име на Джедай и на планета, се обхождат всички планети заедно с населяващите ги планети и ако няма такава планета, то се извежда съобщение, че планета с това име не съществува. В случай, че на тази планета няма Джедай с въведеното от потребителя име, то отново нужното съобщение се показва на екрана. Ако всички тези проверки са минати успешно, то този Джедай се премахва от съответната планета и се изписва съобщението Джедай с кое име е премахнат от коя планета.

```
for (Planet planet : Planet.listOfPlanets) {
   if (planetName.equalsIgnoreCase(planet.getName())) {
       for (Jedi jediOnPlanet : planet.getJedis()) {
           if (jediOnPlanet.name.equalsIgnoreCase(jediName)) {
               planet.getJedis().remove(jediOnPlanet);
               return "Removed " + jediName + " from " + planetName;
           }
       }
       return "There is no jedi named: " + jediName + " on " + planetName;
   }
}
return "There is no planet named: " + planetName;
```

За получаване на Джедая с най-голяма сила е нужно отново да се обходи целия списък и да се сравнят всички сили на Джедаите населяващи избраната планета. Отново, ако посочената планета не е открита чрез създадената булева променлива се отразява това и се показва необходимото съобщение на екрана.

```
for (Planet planet : Planet.listOfPlanets) {
    if (planet.getName().equalsIgnoreCase(planetName)) {
        planetFound = true; // Set the flag to true since planet is found
        for (Jedi jediOnPlanet : planet.getJedis()) {
            if (jediOnPlanet.getStrength() > highestStrength) {
                highestStrength = jediOnPlanet.getStrength();
                strongestJedi = jediOnPlanet;
            }
        }
    }
}
```

За получаване на най-младия Джедай е необходимо потребителят да въведе планетата и ранга, които трябва да притежава като свойство този Джедай. Тук обаче има и допълнително изискване в условието, а именно ако има повече от един най-млад Джедай, то тогава нека се изведе първият по азбучен ред. Това се постига като се създаде един списък с най-младите Джедаи наименуван *'youngestJedis'* (Фиг.9). Статичния метод *'Collections.min'* от класа *'java.util.Collections'* открива минималният елемент в дадена колекция според създадения и зададен компаратор, който в този случай сравнява имената на героите без да се взима под внимание размера на буквите (Фиг.10).

```
if (jediOnPlanet.getRank() == rank) {
    rankFound = true;
    if (jediOnPlanet.getAge() <= youngestAge) {
        youngestJedis.add(jediOnPlanet);
        youngestAge = jediOnPlanet.getAge();
    }
}
```

Фиг. 9

```
Jedi youngestJedi = Collections.min(youngestJedis,
Comparator.comparing(Jedi::getName, String.CASE_INSENSITIVE_ORDER));
builder.append("Youngest Jedi with the specified rank: " +
youngestJedi.toString());
```

Фиг. 10

Най-разпространеният цвят на светлинния меч сред Джедаите на дадена планета и съответно подадения ранг е следващата команда, която ще бъде разгледана в документацията. Тази команда се постига чрез обхождане на планетата и записване на всеки един срещнат цвят в новосъздадената структура от тип *Map* и по-точно *HashMap* – '*colorCountMap*' (Фиг.11). Тук се запазват като ключ наименованието на цвета, а като стойност броят пъти, в които е срещнат. Причината поради която се използва ***HashMap*** е следната:

- Уникалност на ключовете, което гарантира, че при повторно срещане на даден цвят, то той няма да бъде добавен като нов ключ, а ще се увеличи броячът само.
- Също така използването на *HashMap* пред *Map* е предимство при работа с голям обем данни, тъй като *HashMap* използва *хеш* таблица и позволява много бърз достъп до стойностите чрез *хеширане* на ключовете.

```
    if (jediOnThisPlanet.getRank().equals(rank)) {
        rankFound = true;
        String lightsaberColor = jediOnThisPlanet.getLightsaber();
        colorCountMap.put(lightsaberColor,
    colorCountMap.getOrDefault(lightsaberColor, 0) + 1);
    }
```

Фиг. 11

След това трябва да се открие и коя от всичките стойности е най-висока, за да може да се изведе, но ако в случай, че има два цвята с еднакъв брояч, то и двата ще бъдат изведени на екрана(Фиг.12). Условният оператор проверява за максималния брой срещания на даден цвят и ако текущият брой е по-голям от максималния, то той става максимален и след изчистване на списъка се добавя новият най-често срещан цвят.

```
List<String> mostUsedColors = new ArrayList<>();
int maxCount = 0;
for (Map.Entry<String, Integer> entry : colorCountMap.entrySet()) {
    int count = entry.getValue();
    if (count > maxCount) {
        maxCount = count;
        mostUsedColors.clear();
        mostUsedColors.add(entry.getKey());
    } else if (count == maxCount) {
        mostUsedColors.add(entry.getKey());
    }
}
result.append("Most used lightsaber color/s: " + mostUsedColors);
```

Фиг. 12

Почти подобна е и командата за най-често използван цвят на светлинния лъч на Джедаите, който се използва от поне един с ранг *'GRAND MASTER'*. Тук единствената разлика е че се добавя още една булева променлива за намерен Джедай с този специфичен ранг. Създава се един списък с цветовете, които използват *'GRAND_MASTER'* героите. И на база на този списък се прави сравнение дали най-срещаният цвят по принцип, се съдържа и в списъка най-високопоставените в ранглистата Джедаи.

```
for (Map.Entry<String, Integer> entry : colorCountMap.entrySet()) {
    if (entry.getValue() >= maxCount &&
grandMasterColors.contains(entry.getKey())) {
        maxCount = entry.getValue();
        mostUsedColors.add(entry.getKey());
}
```

За извеждане на информацията за дадена планета се създава методът наименуван ‘*print_planet’*. Тук изискването е да се покажат и всички населяващи тази планета Джедаи сортирани по два ключа.

Първият начин е по нарастващ ред на ранга на героите. Този метод се реализира в класа ‘Planet’, като първо от потребителя се изисква въвеждане на името на планетата, чиито Джедаи иска да види. Чрез обхождане на всички планети първо се открива търсената планета, а сортиране се извършва чрез сравнение на ранговете на Джедаите. Важно е да се отбележи, че ‘*Collections.sort’* се използва за сортиране на списъци в Java. Подаден *е ‘Comparator’* обект, който имплементира метода ‘*compare’*. Този метод се извиква, когато трябва да се определи как точно да бъдат сравнени два обекта от класа *Jedi*.
В метода *‘compare’* се сравняват ранговете на двата *Jedi* обекта, *jedi1* и *jedi2*, чрез извикване на метода ‘*compareTo*’ върху техните рангове. Този метод сравнява два ранга и връща:

- положително число, ако *jedi1* има по-висок ранг от *jedi2*.
- отрицателно число, ако *jedi1* има по-нисък ранг от *jedi2*.
- нула, ако ранговете на *jedi1* и *jedi2* са еднакви.

По този начин след края на сортирането вече всички Джедаи, населяващи дадената планета вече героите са подредени във възходящ ред.

```
Collections.sort(planet.getJedis(), new Comparator<Jedi>() {
    @Override
    public int compare(Jedi jedi1, Jedi jedi2) {
        return jedi1.getRank().compareTo(jedi2.getRank());
    }
});
```

Вторият ключ за извеждане на Джедаите е според техните има, сортирани в лексикографска подредба. Тук се създава нов метод, тъй като още един метод изисква подреждане по този начин. Методът сортира низовете, като първо сравнява текстовите части и след това числовите части, ако текстовите части са еднакви.

```
        int result = parts1[0].compareTo(parts2[0]);
if (result == 0 && parts1.length > 1 && parts2.length > 1) {
    int number1 = Integer.parseInt(parts1[1]);
    int number2 = Integer.parseInt(parts2[1]);
    result = Integer.compare(number1, number2);
}
```

Следващата команда за принтиране на данни от Вселената е тази за Джедаите – *'print_jedi'*. Тук изискването е извеждане и на планетата, която Джедаят населява. Като решение се обхожда отново целият списък с планетите и когато Джедай със същото име, което и потребителят е въвел от стандартния вход. Методът се реализира в класа '*Jedi*'.

```
for (Planet planet : Planet.listOfPlanets) {
    for (Jedi jediOnPlanet : planet.getJedis()) {
        if (jediOnPlanet.getName().equalsIgnoreCase(name)) {
            result = jediOnPlanet + ", planet: " + planet.getName();
            return result;
        }
    }
}
```

Последният метод служи за извеждане на информацията за всички Джедаи, населяващи двете планети, които са избрани от потребителя, като тук изискването е отново героите да са сортирани в лексикографски вид. Чрез две кратки условни оператори се проверява за наличието на тези 2 планети и ако и двете условия са изпълнени, то тогава в списъка наречен *'allJedis'* се добавят всички населяващи ги Джедаи чрез метода *'addAll'*.

И тук отново се извиква методът за сортиране по лексикографска подредба.

```
Collections.sort(allJedis, new Comparator<Jedi>() {
    @Override
    public int compare(Jedi jedi1, Jedi jedi2) {
        return compareJediNamesLexicographically(jedi1, jedi2);
    }
});
```

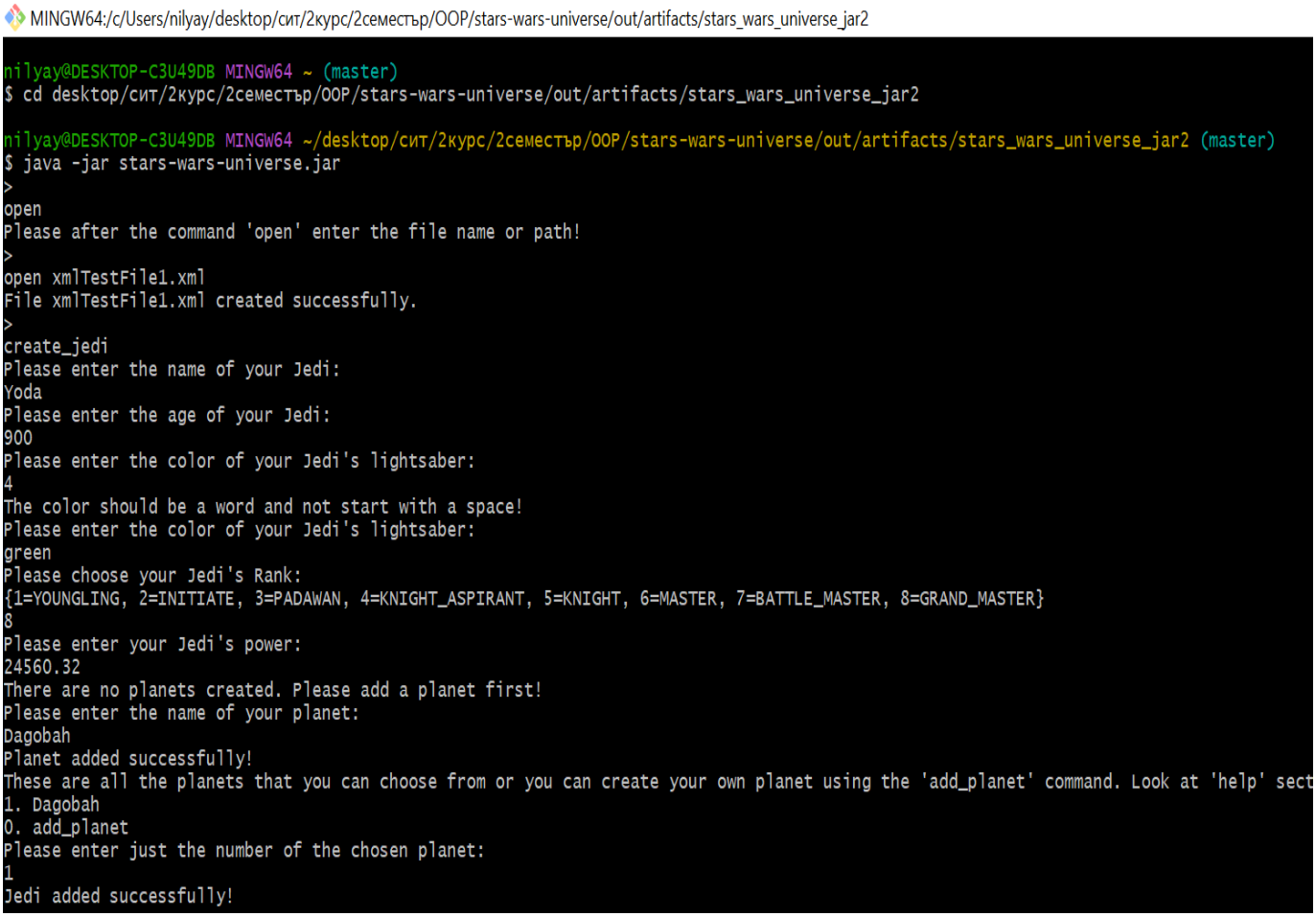
2. Тестване и примерни файлове.

Създаден е примерен файл наименуван *‘xmlTestFile1.xml’*, като приложението има възможността и да създава нови файлове по време на работа, когато клиентът посочи име на файл, което не е намерено.

След посочване на директорията на проекта следва да се изпълни *‘jar‘* файлът. След отварянето на създадения файл той се зарежда в хранилището, но и също така позволява на потребителя да види всички данни, които са запаметени в него, за да може той лесно да работи с тях.

Създаването на Джедай става чрез използването на командата ‘*create_jedi’*, който метод насочва потребителя да въведе необходимите данни, докато всяко едно от тях се валидира на заден план, за да бъдат коректни данните.

В края на изпълненито на опцията се извежда съоябещението,че Джедаят бил добавен успешно.



Препоръчително е след създаването на Джедай, ако потребителят не иска да изгуби въведените данни, да ги запише използвайки командата за запазване ‘*save’*. Това ще запази промените в текущия файл.

```
>
save
Data saved successfully!
```

Повишаването на ранга на Джедай се изпълнява след избора на командата ‘*promote_jedi’*. Тук от потребителя се иска въвеждането на името на Джедая, както и числото, спрямо което ще се изчисли формулата за повишаване и на силата му.

```
promote_jedi
Please enter the name of the Jedi you want to promote:
shaak ti
Found shaak ti
Please enter the multiplier you would like to promote/demote your Jedi's strength:
3.2
Jedi named shaak ti rank:  BATTLE_MASTER
Strength: 62161.68
```

Ако Джедаят вече е постигнал най-високият ранг, то на екрана се изписва съобщението, че неговият ранг е максималният и няма как да се повиши повече.

```
promote_jedi
Please enter the name of the Jedi you want to promote:
yoda
Found yoda
Please enter the multiplier you would like to promote/demote your Jedi's strength:
4.2
Jedi has already achieved the highest rank.
Jedi named yoda rank:  GRAND_MASTER
Strength: 24560.32
```

Понижаването на ранга на Джедая се изпълнява по същият начин, като тук отново, ако Джедаят е достигнал минималния ранг, то той не може да се понижи повече в ранглистата.

```
>
demote_jedi
Please enter the name of the Jedi you want to demote:
dooku
Found dooku
Please enter the multiplier you would like to promote/demote your Jedi's strength:
1.1
Jedi named dooku rank: KNIGHT
Strength: -2030.04
```

```
demote_jedi
Please enter the name of the Jedi you want to demote:
sors bandeam
Found sors bandeam
Please enter the multiplier you would like to promote/demote your Jedi's strength:
3.9
Jedi cannot be demoted further.
Jedi named sors bandeam rank: YOUNGLING
Strength: 290.3
```

Най-младият Джедай се извежда на база на избора на потребителя относно планетата и ранга на Джедая. Съответно извежда се Джедаят с най-малка възраст в тази планета с този ранг. Ако няма такъв Джедай, то се извежда подходящо съобщение, а ако са повече от един на същата възраст се извежда първия по азбучен ред.

```
>
get_youngest_jedi
These are all the planets that you can choose from or you can create your own planet using the 'add_planet' command. Look at 'help' s
1. Dagobah
2. Coruscant
3. HaruunKal
0. add_planet
Please enter just the number of the chosen planet:
2
Please choose your Jedi's Rank:
{1=YOUNGLING, 2=INITIATE, 3=PADAWAN, 4=KNIGHT_ASPIRANT, 5=KNIGHT, 6=MASTER, 7=BATTLE_MASTER, 8=GRAND_MASTER}
5
Youngest Jedi with the specified rank: Jedi: name: 'Anakin Skywalker', rank: KNIGHT, age: 22, lightsaber: 'blue', strength: 8709.32
>
```

Ако Aayla Secura се добави към планета номер 2 – Coruscant със същия ранг на текущия най-млад съответно ‘KNIGHT’, то тогава за най-млад герой ще се изведе тя, тъй като нейното име е преди това на Anakin Skywalker по азбучен ред.

```
>
get_youngest_jedi
These are all the planets that you can choose from or you can create your own planet using the 'add_planet' command. Look at 'help' s
1. Dagobah
2. Coruscant
3. HaruunKal
0. add_planet
Please enter just the number of the chosen planet:
2
Please choose your Jedi's Rank:
{1=YOUNGLING, 2=INITIATE, 3=PADAWAN, 4=KNIGHT_ASPIRANT, 5=KNIGHT, 6=MASTER, 7=BATTLE_MASTER, 8=GRAND_MASTER}
5
Youngest Jedi with the specified rank: Jedi: name: 'Aayla Secura', rank: KNIGHT, age: 22, lightsaber: 'pink', strength: 14980.43
```

Най-силният Джедай се намира като се въведе планетата, за която искаме да получим информацията. Този резултат е получен след намаляването на ранга и съответно силата на Dooku, който преди това беше най-силният герой на планетата Coruscant.

```
>
get_strongest_jedi
Please enter the number of the planet:
These are all the planets that you can choose from or you can create your own planet using the 'add_planet' command. Look at 'help'
1. Dagobah
2. Coruscant
3. HaruunKal
0. add_planet
Please enter just the number of the chosen planet:
2
Jedi: name: 'Ki-Adi-Mundi', rank: MASTER, age: 74, lightsaber: 'blue', strength: 16750.85
>
```

За извеждане на най-често срещания цвят на светлинния меч на Джедаите отново е нужно да се изберат 2 категории: планета и ранг. Ако рангът не е намерен, то се извежда необходимото съобщение.

```
get_most_used_saber_color
These are all the planets that you can choose from or you can create your own planet using the 'add_planet' command. Look at 'help'
1. Dagobah
2. Coruscant
get_most_used_saber_color
These are all the planets that you can choose from or you can create your own planet using the 'add_planet' command. Look at 'help'
1. Dagobah
2. Coruscant
3. HaruunKal
0. add_planet
Please enter just the number of the chosen planet:
2
Please choose your Jedi's Rank:
{1=YOUNGLING, 2=INITIATE, 3=PADAWAN, 4=KNIGHT_ASPIRANT, 5=KNIGHT, 6=MASTER, 7=BATTLE_MASTER, 8=GRAND_MASTER}
5
Most used lightsaber color/s: [blue]
```

При съвпадение на избрания ранг се извежда и цветът, който се използва най-много от тези Джедаи, ако те са повече от един за съответния ранг, се извеждат и двата.

Извеждането на най-често използваният цвят на светлинния меч от поне един Джедай с най-висок ранг се постига по същият начин, само че тук не се изисква въвеждане на ранга, а само планетата.

```
get_most_used_saber_color_grand_master
These are all the planets that you can choose from or you can create your own planet using the 'add_planet' command. Look at 'help' s
1. Dagobah
2. Coruscant
3. HaruunKal
0. add_planet
Please enter just the number of the chosen planet:
3
Most used lightsaber color/s used by at least one Grand Master: [amethyst]
```

Ако няма нито един ‘GRAND_MASTER’ на планетата, то на екрана се появява съобщението, че на избраната планета няма Джедай с най-висок ранг.

```
get_most_used_saber_color_grand_master
These are all the planets that you can choose from or you can create your own planet using the 'add_planet' command. Look at 'help'
1. Dagobah
2. Coruscant
3. HaruunKal
0. add_planet
Please enter just the number of the chosen planet:
2
No Grand Master Jedi found on the specified planet.
```

За извеждането на информацията за дадена планета по два ключа: първо по ранг на Джедаите и след това по имена, сортирани в лексикографски ред се извиква командата *‘print_planet’*.

```
print_planet
[planet name: Dagobah, planet name: Coruscant, planet name: HaruunKal]
Please enter the name of the planet you would like to see the jedis on:
Coruscant
Sorted by Rank:
Jedi: name: 'Obi-Wan Kenobi', rank: KNIGHT, age: 57, lightsaber: 'blue', strength: 14750.2
Jedi: name: 'Anakin Skywalker', rank: KNIGHT, age: 22, lightsaber: 'blue', strength: 8709.32
Jedi: name: 'Ki-Adi-Mundi', rank: MASTER, age: 74, lightsaber: 'blue', strength: 16750.85
Jedi: name: 'Dooku', rank: MASTER, age: 83, lightsaber: 'blue', strength: 20300.41

Sorted by Name:
Jedi: name: 'Anakin Skywalker', rank: KNIGHT, age: 22, lightsaber: 'blue', strength: 8709.32
Jedi: name: 'Dooku', rank: MASTER, age: 83, lightsaber: 'blue', strength: 20300.41
Jedi: name: 'Ki-Adi-Mundi', rank: MASTER, age: 74, lightsaber: 'blue', strength: 16750.85
Jedi: name: 'Obi-Wan Kenobi', rank: KNIGHT, age: 57, lightsaber: 'blue', strength: 14750.2
```

Следващата команда е за извеждане на информацията за избрания Джедай, като е нужно потребителят единствено да въведе името му.

```
print_jedi
Please enter the name of a jedi you would like to see information about:
Shaak Ti
Jedi: name: 'Shaak Ti', rank: MASTER, age: 40, lightsaber: 'blue', strength: 14800.4, planet: HaruunKal
```

Последната команда е за извеждане на информация за населяващите двете, избрани от потребителя, герои. Нужно е да се въведат от стандартния вход имената на планетите със знак „плюс“ (+) между тях и интервал.

```
dagobah + coruscant
Jedi: name: 'Anakin Skywalker', rank: KNIGHT, age: 22, lightsaber: 'blue', strength: 8709.32
Jedi: name: 'Dooku', rank: MASTER, age: 83, lightsaber: 'blue', strength: 20300.41
Jedi: name: 'Ki-Adi-Mundi', rank: MASTER, age: 74, lightsaber: 'blue', strength: 16750.85
Jedi: name: 'Luke Skywalker', rank: KNIGHT, age: 53, lightsaber: 'green', strength: 17659.99
Jedi: name: 'Obi-Wan Kenobi', rank: KNIGHT, age: 57, lightsaber: 'blue', strength: 14750.2
Jedi: name: 'Plo Koon', rank: MASTER, age: 385, lightsaber: 'blue', strength: 19450.5
Jedi: name: 'Yoda', rank: GRAND_MASTER, age: 900, lightsaber: 'green', strength: 24560.32
```

Запазването на файла в друга директория или в друг файл се изпълнява чрез командата *‘saveas’*, като приложението ще изисква въвеждане на абсолютния път на файла.

```
saveas
Please enter the path you would like to save your file:
C:/Users/nilyay/Desktop/сит/2курс/2семестър/OOP/testSave.xml
Data saved successfully!
>
```

Опцията за помощ извежда на екрана следната информация:

```
help
These are all the commands you can run:
open (file) -> open a file after entering its name
close -> closes currently opened file
save -> saves the currently opened file
saveas -> saves the currently opened file in a directory or another file
help -> prints commands information
add_planet -> add a new planet
create_jedi -> create a new Jedi
remove_jedi -> remove a jedi from a planet
promote_jedi -> let the jedi get better and promote his rank and strength
demote_jedi -> demote the jedi a level down on the rank list and lower the strength
get_strongest_jedi -> get the strongest jedi of all Jedis on the planet
get_youngest_jedi -> get the youngest jedi on the planet with a specified by you rank
get_most_used_saber_color -> get the most used saber color by jedis on the planet and with a specified by you rank
get_most_used_saber_color_grand_master -> get the most used saber color used from at least one Grand Master
print_planet -> print jedis on a planet sorted by rank and then lexicographically
print_jedi -> print jedi details and on which planet it is located
planetName + planetName -> get the jedis on two planets sorted lexicographically
exit -> exits the program
```

Докато тази за изход излиза от системата.

```
>
exit
Exiting the system!

nilyay@DESKTOP-C3U49DB MINGW64 ~/desktop/сит/2курс/2семестър/OOP/stars-wars-universe/out/artifacts/stars_wars_universe_jar2 (master)
```

5. Заключение

1. **Обобщение на изпълнението на началните цели.**

Създадох приложение с малка Вселена, вдъхновена от поредицата „Междузвездни войни“. Програмата се изпълнява в Командния ред като се стартира създаденият ‘*jar*’ файл. Целите, които и в края на проекта са изпълнени са работа с *XML* файлове, като действията са отваряне, затваряне, запазване, запазване като и меню помощ. Контекстно свързани с темата на проекта са и командите за добавяне на планета и населяването и с нови Джедаи, като с героите има възможност за малко повече действия. Например могат да бъдат повишени или понижени в ранглистата. Някои от командите позволят извличане на данни от специфични планети на Джедаи със специфични рангове.

2. **Насоки за бъдещо развитие и усъвършенстване.**

В бъдеще при развиване на приложението е нужно да се помисли върху няколко неща, които в този момент на реализиране на проекта не бяха изпълнени.

- При добавяне на по-голям брой опции за избор от менюто, трябва да се използва различна структура от тази на условния блок със селектор, тъй като този начин няма да бъде най-оптималният и ще изисква писане на повече код, който ще бъде труден за разбиране.
- Също така при обхождане на списъка на планетите и населяващите я Джедаи се използват обикновен цикъл от тип ‘for’, което при по-голям размер на данните, ще затрудни изпълнението на програмата заради забавяне.
- Откриването на грешки също е проблем, който според мен трябва да се доразвие, за да може да се дава нужната информация на потребителя, която да му помогне да отстрани грешките.
- Допълнителни опции към менюто също могат да се добавят, за да може потребителят да има достъп да по-голяма функционалност и да може да има повече функции, които да изпълнява.

Източници:

- Regex: https://www.javatpoint.com/java-regex
- Comparator: https://www.geeksforgeeks.org/comparator-interface-java/
- Decimal Format: https://stackoverflow.com/questions/433958/java-decimal-formatting-using-string-for
- Ordinal for Enum: https://www.javatpoint.com/post/java-enum-ordinal-method

Връзка към хранилището: https://github.com/nilyay/star-wars-universe/tree/master/out