Изисквания

Курсов проект Обектно-ориентирано програмиране 1 част за специалност Софтуерни и Интернет технологии, Киберсигурност

Обща информация за проектите

Проектът е цялостна задача, която трябва да решите с помощта на познанията по дисциплината Обектно-ориентирано програмиране 1 част. Инструмент за реализация - Java

1. Срок за предаване на **завършените** проекти: 13 седмица
2. По време на работата по проекта трябва да се използва хранилище в системата Github или GitLab.
    - Хранилището трябва да се обновява регулярно (всяка седмица), паралелно с работата ви по проекта.
    - Направените от вас междинни промени ще участват в оценяването на крайния резултат.
3. В обявения срок трябва да предадете:
    - Документация на проекта;
    - Изходен код на решението;
    - JavaDoc
    - За защитата трябва да имате няколко примера, подбрани от Вас, които демонстрират работата на задачата.
4. Предаването става чрез прикачване на ZIP архив към съответното задание в MS Teams, който съдържа всички файлове, необходими за компилирането на проекта, отделен ZIP архив с JavaDoc и отделен файл извън архива с документацията към проекта.
5. Документацията на проекта трябва да съдържа:
    - Анализ на задачата и Вашия подход за решение (на какви стъпки сте разделили решението, какъв метод или алгоритъм сте избрали, как сте се справили с конкретен проблем);
    - Кратко описание на класовете, създадени от Вас за решение на задачата - трябва да се направи със система за генериране на документация (JavaDoc) от коментари в изходния код (описание на класовете и начинът им на използване).
    - Връзка към вашето хранилище в Github (GitLab);
    - Съдържание на документацията е описано по-долу.
6. По време на защитата трябва да разкажете в рамките на 10 минути Вашето решение и да демонстрирате работата на програмата с подготвени от Вас данни.
7. По време на защитата се очаква да можете да отговорите на различни въпроси, като например: (1) дали сте обмислили други варианти на реализация и ако да — какви,

(2) как точно работят различните части от вашия код и какво се случва на по-ниско ниво и др.

8. Възможно е даването на малка задача за допълнение или промяна на функционалността на проекта ви, която вие трябва да реализирате на място.
9. Невъзможност да реализирате малката задача на място означава, че не познавате добре проекта си и поражда съмнения, че сте ползвали чужда помощ за реализацията му. Последното ще се отрази негативно на оценката ви.
10. Установено плагиатство на код от колеги и от други източници води до анулиране на работата и оценка Слаб 2. За плагиатство се счита използване на код в решението, чиито източник не е изрично упоменат.

Структура на документацията

Заглавна страница

Задание на курсовия проект

Съдържание

Глава 1. Увод

1.1. Описание и идея на проекта

1.2. Цел и задачи на разработката

Глава 2. Преглед на предметната област

2.1. Основни концепции и алгоритми, които ще бъдат използвани

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

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

Глава 3. Проектиране

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

3.2. Диаграми/Блок схеми (на структура и поведение - по обекти, алгоритми за изпълнение на командите от заданието)

Глава 4. Реализация

4.1. Реализация на класове (включва реализацията на функционалностите с техните класове и малки фрагменти от кода за важните функционалности)

4.1.1 Алгоритми и оптимизации (слоеве с най-важните извадки от кода).

Глава 5. Тестване

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

Глава 6. Заключение

6.1. Обобщение на изпълнението на началните цели

Използвана литература

Изисквания за оформяне на документацията на проекта:

1. Шаблонът е препоръчителен и може да се променя в зависимост от конкретното задание.
2. Йерархията на структуриране на съдържанието да не бъде повече от 3 нива, номерирани с арабски цифри – напр. 1.2.3.
3. Чуждестранните термини да бъдат преведени, а където това не е възможно – цитирани в *курсив* и нечленувани.
4. Страниците да бъдат номерирани с арабски цифри, в долния десен ъгъл.
5. Използваният шрифт за основния текст на описанието да бъде Times New Roman 12 или Arial 10, и Consolas за кода, с междуредие 16pt.
6. Да се избягва пренасянето на нова страница на заглавия на секции, фигури и таблици.
7. Да се избягват празни участъци на страници вследствие пренасянето на фигури на нова страница.
8. Всички фигури и таблици да бъдат номерирани и именовани (непосредствено след фигурата или преди таблицата).
9. Всички фигури и таблици да бъдат цитирани в текста.
10. Всеки термин, дефиниция, алгоритъм или информация, която е взета от литературен източник или Интернет трябва да бъде цитирана. Да се използва БДС стандарта за цитиране.
11. Всички цитати да бъдат отразени в списъка на използваната литература.
12. Всички източници от списъка на използваната литература да бъдат цитирани в текста.
13. Документацията се печата едностранно и се подвързва в папка.

Работа с командния ред (Command line interface)

Вашата програма трябва да позволява на потребителя да отваря файлове (open), да извършва върху тях някакви операции, след което да записва промените обратно в същия файл (save) или в друг, който потребителят посочи (save as). Трябва да има и опция за затваряне на файла, без записване на промените (close). За целта, когато програмата ви се стартира, тя трябва да позволява на потребителя да въвежда команди и след това да ги изпълнява.

Когато отворите даден файл, неговото съдържание трябва да се зареди в паметта, след което файлът се затваря. Всички промени, които потребителят направи след това трябва да се пазят в паметта, но не трябва да се записват обратно, освен ако потребителят изрично не укаже това.

Във всеки от проектите има посочен конкретен файлов формат, с който приложението ви трябва да работи. Това означава, че:

1. то трябва да може да чете произволен валиден файл от въпросния формат;
2. когато записва данните, то трябва да създава валидни файлове във въпросния формат.

Както казахме по-горе, потребителят трябва да може да въвежда команди, чрез които да посочва какво трябва да се направи. Командите могат да имат нула, един или повече параметри, които се изреждат един след друг, разделени с интервали.

Освен ако не е казано друго, всяка от командите извежда съобщение, от което да е ясно дали е успяла и какво е било направено.

Дадените по-долу команди трябва да се поддържат от всеки от проектите. Под всяка от тях е даден пример за нейната работа:

Open

Зарежда съдържанието на даден файл. Ако такъв не съществува се създава нов с празно съдържание.

Всички останали команди могат да се изпълняват само ако има успешно зареден файл.

След като файлът бъде отворен и се прочете, той се затваря и приложението ви вече не трябва да работи с него, освен ако потребителят не поиска да запише обратно направените промени (вижте командата save по-долу), в който случай файлът трябва да се отвори наново. За целта трябва да изберете подходящо представяне на информацията от файла.

Ако при зареждането на данните, приложението ви открие грешка, то трябва да изведе подходящо съобщение за грешка и да прекрати своето изпълнение.

```
> open C:\Temp\file.xml
Successfully opened file.xml
```

Close

Затваря текущо отворения документ. Затварянето изчиства текущо заредената информация и след това програмата не може да изпълнява други команди, освен отваряне на файл (Open).

```
> close
Successfully closed file.xml
```

Save

Записва направените промени обратно в същия файл, от който са били прочетени данните.

```
> save
Successfully saved file.xml
```

Save As

Записва направените промени във файл, като позволява на потребителя да укаже неговия път.

```
> save as "C:\Temp\another file.xml"
Successfully saved another file.xml
```

Help

Извежда кратка информация за поддържаните от програмата команди.

```
> help
The following commands are supported:
open <file> opens <file>
close             closes currently opened file
save              saves the currently open file
cave as <file>    saves the currently open file in <file>
help              prints this information
exit              exists the program
```

Exit

Излиза от програмата

```
> exit
Exiting the program...
```

Проект 16: Star Wars Universe 005.1

Вселената на Star Wars започнала с джедаите, като всеки такъв трябва да притежава:

- ∉ джедайско име
- ∉ ранг, като следните са подредени в нарастващ ред – YOUNGLING, INITIATE, PADAWAN, KNIGHT-ASPIRANT, KNIGHT, MASTER, BATTLE_MASTER и GRAND_MASTER
- ∉ възраст
- ∉ цвят на светлинния меч (символен низ, който се въвежда от клавиатурата)
- ∉ сила (зададена с някакво число от 1 до 2)

След това решил да създаде планетите и луните, като всяка такава има име и джедаи, които я населяват.

Да се изготви приложение, което поддържа следните команди:

`add_planet <planet_name>`	добавя нова планета
`create_jedi <planet_name> <jedi_name> <jedi_rank> <jedi_age> <saber_color> <jedi_strength>`	функцията да извежда съобщение дали добавянето е било успешно или не (съществува джедай с такова име на тази или друга планета, или не съществува планета с такова име).
`removeJedi <jedi_name> <planet_name>`	функцията да извежда съобщение дали премахването е било успешно или не (джедаят не населява тази планета).
`promote_jedi <jedi_name> <multiplier>`	повишава дадения джедай с един ранг нагоре в стълбицата и увеличава силата му по формулата `jedi_strength += (multiplier * jedi_strength)` (не може да се повишава повече от ранг `GRAND_MASTER` и `multiplier` трябва да е положително число от тип `double`)

`demote_jedi <jedi_name> <multiplier>`	намаля ранга на подаденият джедай с един ранг надолу в стълбицата и понижава силата му по формулата `jedi_strength -= (multiplier * jedi_strength)` (не може да се понижава повече от ранг `YOUNGLING` и `multiplier` трябва да е положително число от тип `double`)
`get_strongest_jedi <planet_name>`	извежда информацията за най-силния джедай на подадената планета (с най-голяма сила).
`get_youngest_jedi <planet_name> <jedi_rank>`	извежда най-младия джедай, населяващ подадената планета и е със съответен ранг (**ако са повече от един, да се изведе първият по азбучен ред, ако няма нито един да се изведе подходящо съобщение**)
`get_most_used_saber_color <planet_name> <jedi_rank>`	връща най-разпространения цвят на светлинния меч в подадения ранг на съответната планета
`get_most_used_saber_color <planet_name>`	връща най-разпространения цвят на светлинния меч планета, който се ползва от поне един **GRAND_MASTER**
`print <planet_name>`	извежда по подходящ начин името на планетата и населяващите я джедаи, **сортирани първо в нарастващ ред по ранг, после по втори ключ - лексикографски по името**
`print <jedi_name>`	извежда по подходящ начин информацията за джедаят, както и коя планета населява в момента
`<planet_name> + <planet_name>`	извежда на екрана в сортиран вид (**лексикографски**) информацията за населяващите двете планети джедаи.

Програмата трябва да съхранява информацията планетите и населяващите ги джедаи **във файл** и да се поддържат командите за работа с файлове, описани в секцията **Работа с командния ред. Всички команди са с малки латински букви, а аргументите са разделени с един интервал.**