Теми за проекти

курс Обектно-ориентирано програмиране 1 част проект
за специалност Софтуерни и Интернет технологии
летен семестър 2022/2023 г.

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: 26.05.2023 г.
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;
    - 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 и 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 (избраната архитектура, описание на член-данните и член-функциите на класовете и начинът им на използване);
    - Идеи за бъдещи подобрения;
    - Връзка към вашето хранилище в Github(GitLab);
    - Примерно съдържание на документацията е описано по-долу.
    - Описанието на класовете трябва да се направи със система за генериране на документация от коментари в изходния код. При използване на такава система отпада изискването да се предава текстовата документация, описана по-горе, но се изисква описание в коментарите на кода на създадените класове, както и на техните методи и член данни. Като част от

курсовия проект трябва да се предадат и генерираните файлове с документация.

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.

Примерна структура на документацията

Глава 1. Увод (1 стр.)

1.1. Описание и идея на проекта (3–4 изр.)

1.2. Цел и задачи на разработката (½–1 стр.)

1.3. Структура на документацията (3–4 изр.)

Глава 2. Преглед на предметната област (½–1 стр.)

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

2.2. Дефиниране на проблеми и сложност на поставената задача

2.3. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите

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

Глава 3. Проектиране (1–2 стр.)

3.1. Обща структура на проекта пакети който ще се реализират

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

Глава 4. Реализация, тестване (1–2 стр.)

4.1. Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода)

4.2. Алгоритми и Оптимизации.

4.3. Планиране, описание и създаване на тестови сценарии (създаване на примери)

Глава 5. Заключение (2–3 изр.)

5.1. Обобщение на изпълнението на началните цели

5.2. Насоки за бъдещо развитие и усъвършенстване

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 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.
11. Всички цитати да бъдат отразени в списъка на използваната литература.
12. Всички източници от списъка на използваната литература да бъдат цитирани в текста.

Съдържание - примерно

Обща информация за проектите 1

Съдържание 5

Работа с командния ред 7
- Open 7
- Close 8
- Save 8
- Save As 8
- Help 8
- Exit 8

Проект 1: Приложение за работа с електронни таблици 10
- Представяне на данните 10
- Типове данни в таблицата 10
- Нужна функционалност 11
- Извеждане на таблицата на екрана 12
- Редактиране на клетки 12
- Формули 12

Проект 2: Работа със SVG файлове 14
- Операции 15
- Примерен SVG файл figures.svg 16

Проект 3: XML Parser 18

Проект 4: Недетерминиран краен автомат 20

Проект 5: Контекстно-свободна граматика 21

Проект 6: Бази от данни 22
- Поддържани типове данни 22

Проект 7: Traveller’s app **Error! Bookmark not defined.**

Проект 8: Джурасик парк **Error! Bookmark not defined.**

Проект 9: Големи числа **Error! Bookmark not defined.**

Проект 10: Библиотека 25

Проект 11: Хотел 28

Проект 12: Склад 30

Проект 13: СУСИ 32

Проект 14: Билети 35

Проект 15: Личен календар 37

Проект 16: JSON парсер 39

Проект 17: Растерна графика 40

Проект 18: Dungeons & Dragons **Error! Bookmark not defined.**

Проект 19: Star Wars Universe 0.1 43

Проект 20: Шах **Error! Bookmark not defined.**

Бонус проекти: игри 47

- Snake 47
- Alien Attack 47
- Sokoban 48
- Tetris 48
- Breakout 48
- Xonix 48
- Pacman 49
- Mine Sweeper 49
- Pong 49
- Lander 49
- Arcade Volleyball 50
- Frogger 50
- The Game of Life 50
- 2048 51
- Занимателна математика 51
- Проект 18: Dungeons & Dragons 52
- Проект 20: Шах 52

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.

```
> saveas "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
saveas <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 19: Star Wars Universe 0.1

Ванката е голям фен на Star Wars, затова решил да си направи малко проектче по ООП на тази тематика. За целта той иска да пресъздаде вселената от поредицата.

Първо той започнал с джедаите, като решил че 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 тип double)

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 джедая, както и коя планета населява в момента
`<planet_name> + <planet_name>`	извежда на екрана в сортиран вид (**лексикографски**) информацията за населяващите двете планети джедаи.

Ванката искал освен това да може да запазва информацията, която е вкарал в програмата си за после и да може да я зарежда наново. Тоест 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.**