

Теми за проекти

*курс Обектно-ориентирано програмиране 1 част проект
за специалност Софтуерни и Интернет технологии
летен семестър 2021/2022 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: 29.05.2022 г.
2. По време на работата по проекта трябва да се използва хранилище в системата Github или GitLab.
 - Хранилището трябва да се обновява регулярно, 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, избрани 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

курсoвия 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.

Работа с командния ред

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...
```

Бази от данни

Да се реализира програма, поддържаща операции с прости бази от данни. Базите данни се състоят от серии от таблици, като всяка таблица е записана в собствен файл. Базата данни е записана в главен файл (каталог), които съдържа списък от таблиците в базата данни, като за всяка таблица е зададено име и файл, в който таблицата е записана.

Поддържани типове данни

Всяка “колона” на таблица в базата данни има тип, като в една таблица може да има едновременно колони от различни типове. Вашето приложение трябва да може да поддържа следните типове:

Цяло число – поредица от цифри, без никакви други символи между тях. В началото на числото може да има знак '+' или '-'. Например:

```
123
-123
+123
```

Дробно число – поредица от цифри, следвана от символ за точка и след нея друга поредица от цифри. В началото на числото може да има знак '+' или '-'. Например:

```
123.456
-123.456
+123.456
```

Символен низ (стринг) – поредица от произволни символи оградени в кавички. Подобно на низовете в C++, ако искате да включите символа за кавичка в даден низ, трябва да го представите като \", а ако искате да включите наклонена черта, трябва да я представите като \\. Например:

```
"Hello world!"
"C:\\temp\\"
"\\"This is a quotation\\""
```

Освен конкретна стойност, дадена клетка в даден ред на таблицата може да е “празна”. Такива клетки да се обозначават специално и да е изписват като “NULL”.

След като приложението отвори даден файл, то трябва да може да извършва посочените по-долу операции, в допълнение на общите операции (close, save, save as, help и exit):

import <file name>	Добавя в базата данни нова таблица от файл. Във файла е записана информация за типа на всяка колона. Всяка таблица има име. При опит за зареждане на таблица с име, което съвпада с името на някоя вече заредена таблица, системата да дава грешка. Добавената таблица се записва в каталога на базата от данни.
showtables	Показва списък с имената на всички заредени таблици
describe <name>	Показва информация за типовете на колоните на дадена таблица
print <name>	Показва всички редове от дадена таблица. Да се реализира диалогов режим, позволяващ съдържанието на таблицата да се преглежда по страници (такива, че се събират на един екран) със следните команди: следваща страница, предишна страница, изход.
export <name> <file name>	Записва таблица във файл
select <column-n> <value> <table name>	Извежда всички редове от таблицата, които съдържат стойността "value" в клетката с дадения пореден номер. Да се реализира извеждане по страници
addcolumn <table name> <column name> <column type>	Добавя нова колона (с най-голям номер) в дадена таблица. За всички съществуващи редове от таблицата, стойността на тази колона да е празна.
update <table name> <search column n> <search value> <target column n> <target value>	За всички редове в таблицата, чиято колона с пореден номер <search column n> съдържа стойността <search column value> се променят така, че колоната им с пореден номер <target column n> да получи стойност <target value>. Да се поддържа стойност NULL.
delete <table name> <search column n> <search value>	Изтрива всички редове в таблицата, чиято колона <search column n> съдържа стойността <search column value>
insert <table name> <column 1> ... <column n>	Вмъква нов ред в таблицата със съответните стойности
innerjoin <table 1> <column n1> <table 2> <column n2>	Извършва операцията Inner Join над две таблици спрямо колоните <column n1> в първата таблица и <column n2> във втората. Създава нова таблица и извежда идентификатора ѝ.
rename <old name> <new	Преименува таблица. Отпечатва грешка, ако новото

name>	име не е уникално.
count <table name> <search column n> <search value>	Намира броя на редовете в таблицата, чиито колони съдържат дадената стойност
aggregate <table name> <search column n> <search value> <target column n> <operation>	Извършва дадена операция върху стойностите от колоната <target column n> на всички редове, чиито колони с номер <search column n> съдържат стойността <search value>. Възможните операции са sum, product, maximum, minimum. Системата да дава грешка, ако колоните не са числови.