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 1: Приложение за работа с електронни таблици

Представяне на данните

Данните на една таблица ще записваме в текстов файл по следния начин:

1. Всеки ред във файла представя отделен ред в таблицата.
2. Всеки ред във файла съдържа данни разделени със запетаи. Тези данни се интерпретират като стойностите в клетките на реда.
3. Всеки ред в таблицата може да съдържа различен брой клетки. Затова и всеки ред във файла може да съдържа различен брой елементи разделени със запетаи.
4. Празен ред във файла представя празен ред в таблицата. (т.е. ред, в който всички клетки са празни).
5. Между две запетаи във файла може да няма никакви данни. По този начин се представя празна клетка.
6. Между данните и запетаите може да има произволен брой празни символи (whitespace).

Така за една таблица може да има различни представяния. Например таблицата:

10	20	30	40
10		1000	
	10		

може да се представи по следните начини (възможни са и други представяния):

```
10, 20, 30, 40

10,,1000,
,,,
,10
```

```
10, 20 , 30  , 40

10,   , 1000,
  ,    ,    ,
  , 10
```

Типове данни в таблицата

Всяка клетка в таблицата има тип, като в една таблица може да има едновременно клетки 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 '-'I. 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\""
```

Формула – формулата винаги започва със символ за равенство. В нея могат да участват следните операции: събиране (+), изваждане (-), умножение (*), деление (/) и степенуване (^). Във формулата могат да участват или числа или препратки към клетки в таблицата. Ако във формулата участва препратка към клетка, на това място в изчислението трябва да се използва стойността съхранена в дадената клетка. Повече информация за формулите е дадена по-долу.

Нужна функционалност

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

`print`	Извежда съдържанието на таблицата на екрана
`edit`	Редактира съдържанието на дадена клетка. За целта потребителят въвежда текст, който ще бъде новото съдържание на клетката. Забележете, че по този начин може да се промени типът на дадена клетка, например от число, тя може да стане формула.

Както беше казано в общата за всички проекти информация, ако при зареждането на данните, приложението ви открие грешка, то трябва да изведе подходящо съобщение за грешка и да прекрати своето изпълнение. Съобщението трябва да подскаже на потребителя какво не е наред във входните данни. Например:

- Ако липсва запетая трябва да се изведе на кой ред и след кой символ липсва запетаята;
- Ако съдържанието на дадена клетка е от неизвестен тип, трябва да се изведе на кой ред и коя колона е клетката и какво точно е некоректното съдържание. Например нека предположим, че на ред 2, колона 5, потребителят е въвел 123.123.123. Приложението ви може да изведе например следното съобщение: *“Error: row 2, col 5, 123.123.123 is unknown data type”*.

Извеждане на таблицата на екрана

При извеждане на заредената таблица (командата print), данните в колоните трябва да се подравнят. Между отделните колони трябва да се поставят символи за отвесна черта (|). По-долу е даден пример за входен файл и възможно негово извеждане:

Входен файл	Извеждане

`10, "Hello world!", 123.56`
`"\"Quoted\""`
`1, 2, 3, 4`
	`10       | Hello world! | 123.56 |   |`
	`"Quoted" |              |        |   |`
	`1        |            2 |      3 | 4 |`

Редактиране на клетки

Командата Edit трябва да позволява (с подходящи параметри) на потребителя да променя стойностите на отделните клетки. Това става като се укажат реда и колоната на клетката, която искаме да променим, а също и каква стойност да запише в нея. Потребителят може да въведе произволен тип данни, който се поддържа от вашата програма (например цяло число, дробно число, низ, формула и т.н.).

Ако потребителят въведе неправилни данни, приложението ви не трябва да променя нищо в таблицата, а само да изведе на екрана съобщение, че са въведени неправилни данни. В този случай приложението ви НЕ трябва да прекратява своето изпълнение.

Формули

Номерата на редовете и клетките в таблицата започват от 1. Препратка към ред <N> и колона <M> в таблицата се записва така: R<N>C<M>. Например клетката в ред 10 и колона 5 се представя като R10C5.

В дадена формула могат да участват единствено:

1. Литерали: цели или дробни числа.
2. Препратки към произволни типове клетки.

При сметките важат следните правила:

1. Ако в дадена формула участват само числа, то сметката се извършва по традиционните правила на аритметиката. Като специален случай можем да отделим делението на две цели числа. В такъв случай не бива да губите остатъка и резултатът трябва да бъде дробно число (например 1 делено на 2 дава резултат 0,5).
2. Ако в дадена формула участва низ, той трябва да се конвертира до число. Това става по следния начин: Ако низът съдържа само цифри или поредица от цифри, символ точка и друга поредица от цифри, той се конвертира до съответното число. Всички други низове се конвертират до нула. Например:

Низ	Конвертирана стойност
“123”	123
“123.456.789”	0
“123.456”	123.456
“Hello world”	0
“123abc”	0

3. Ако в дадена формула участва празна клетка, тя се конвертира до нула. Това важи и за клетки, чиито координати надхвърлят размерите на таблицата.
4. Ако в дадена формула има грешка (например деление на нула), приложението ви не трябва да прекъсва своето изпълнение. Вместо това, когато то извежда таблицата на екрана, в съответната клетка се извежда ERROR, вместо получен резултат.

По-долу е дадена примерна таблица. В нея клетките в жълт цвят са от тип число. Клетките в зелено са от тип символен низ:

	Колона 1	**Колона 2**	**Колона 3**
Ред 1	10	Hello world!	123.56

Ред 2	123		

По-долу са дадени формули, които се оценяват в примерната таблица по-горе. За всяка формула е дадена и нейната оценка:

Формула в клетката	**Реално извършена сметка**	**Стойност на клетката**	**Коментар**
= 10 + 10	10 + 10	20	
= R1C1 + R1C3	10 + 123.56	133.56	
= R1C1 * R1C2	10 * 0	0	Низът „Hello world!“ се конвертира до нула
= R1C1 * R2C1	10 * 123	1230	Низът „123“ се конвертира до 123 Клетката на ред 2, колона 2 е празна В таблицата няма ред 200 и колона 200. Считаме, че тя е празна. т „123“ се конвертира до 123
= R1C1 * R2C2	10 * 0	0	Клетката на ред 2, колона 2 е празна
= R1C1 * R200C1	10 * 0	0	В таблицата няма ред 200 и колона 200. Считаме, че тя е празна.
= 10 / 0	10 / 0	ERROR	
= 10 / R1C2	10 / 0	ERROR	
= R1C1 / R1C2	10 / 0	ERROR	