Документация

към проект по Обектно-ориентирано програмиране

на тема „Работа със SVG файлове“

* Имена: Дана Петрова Бъчварова
* Специалност: Компютърни науки
* Курс: 1
* Поток: 1
* Факултетен номер: 4MI0800427

Съдържание

[Глава 1. Увод 3](#_Toc168254639)

[Глава 2. Преглед на предметната област 3](#_Toc168254640)

[2.1. Основни дефиниции и концепции 3](#_Toc168254641)

[2.1.1. SVG файлове 3](#_Toc168254642)

[2.1.2. Принципи и концепции на обектно-ориентираното програмиране 4](#_Toc168254643)

[2.2. Дефиниране на проблеми и сложност на поставената задача 4](#_Toc168254644)

[2.2.1. Лесни за разрешаване подзадачи 4](#_Toc168254645)

[2.2.2. Средно трудни за разрешаване подзадачи 4](#_Toc168254646)

[2.2.3. Трудни за разрешаване подзадачи 4](#_Toc168254647)

[Глава 3. Проектиране и архитектура 5](#_Toc168254648)

[3.1. Фигури 5](#_Toc168254649)

[3.2. Основни методи за работа с файлове и менажиране на колекции от фигури 5](#_Toc168254650)

[3.3. Методи за приемане на команди от командния ред, валидиране и екзекутиране на програмата 6](#_Toc168254651)

[3.4. Помощни класове и структури 6](#_Toc168254652)

[Глава 4. Реализация и тестване 6](#_Toc168254653)

[4.1. Управление на паметта и алгоритми 6](#_Toc168254654)

[4.2. Тестове 6](#_Toc168254655)

[4.3. Използвани помощни инструменти и библиотеки 6](#_Toc168254656)

[Глава 5. Заключение 7](#_Toc168254657)

[5.1. Изпълнение на началните цели 7](#_Toc168254658)

[5.2. Бъдещо развитие и усъвършенстване 7](#_Toc168254659)

[Примерно използване на приложението: 8](#_Toc168254660)

[GitHub Repository 9](#_Toc168254661)

# Глава 1. Увод

Представяният проект представлява приложение, което обработва SVG (Scalable Vector Graphics) файлове чрез въвеждане на команди в командния ред. Основната му цел е да предостави на потребителите възможността да менажират SVG файлове и фигурите, които те съдържат, по ефективен и интуитивен начин. Това приложение може да бъде разглеждано като един вид SVG *parser*, който поддържа работа с краен брой базови SVG фигури. Основните му функционалности са:

* **Отваряне и четене на поддържаните фигури от файл** – Потребителите могат да зареждат съществуващи SVG файлове и да извличат информация за поддържаните фигури, съдържащи се в тях. Това включва четене на атрибути като координати, размери и цветове на фигурите. Приложението анализира SVG съдържанието и предоставя удобен интерфейс за достъп до тази информация.
* **Създаване на нов SVG файл –** Приложението позволява на потребителите да създават нови SVG файлове от нулата. Там те могат да добавят базови фигури като кръг, правоъгълник, линия и други (ако се добавят към поддържаните фигури), задавайки техните атрибути. След като тези фигури са добавени, приложението генерира валиден SVG файл, който може да бъде запазен в посочена директория.
* **Въвеждане и запазване на промени по поддържаните фигури във файл** – Приложението дава възможност на потребителите да правят промени на съществуващи вече SVG файлове. Това включва модифициране на атрибути на поддържаните фигури и транслиране. След като промените са направени, приложението може да запази новия SVG файл, като гарантира, че новото съдържание е валидно и съответства на SVG спецификацията.
* **Извеждане на информация за поддържаните фигури във файл** – Приложението може да генерира отчети за съдържащите се в SVG файл поддържани фигури.

# Глава 2. Преглед на предметната област

## 2.1. Основни дефиниции и концепции

### 2.1.1. SVG файлове

SVG (Scalable Vector Graphics) файлът е XML-базиран формат за описване на двумерна векторна графика. Той е широко използван за представяне на изображения поради своите предимства в мащабирането и резолюцията. Пример за съдържание на SVG файл:

<svg>

    <circle cx="5" cy="5" r="5" fill="green" />

    <rect x="3" y="4" width="10" height="12" fill="blue" />

    <line x1="7" x2="7" y1="10" y2="10" stroke="red" />

</svg>

### 2.1.2. Принципи и концепции на обектно-ориентираното програмиране

При разработването на проекта са следвани основните концепции в обектно-ориентираното програмиране, а именно: енкапсулация, наследяване, полиморфизъм и абстракция:

* **Енкапсулация** – Данните и методите, които се отнасят към една и съща област на функциониране са опаковани заедно в обект. В случая са използвани и сингълтони – шаблони, от които се създава единствен екземпляр.
* **Наследяване** – Благодарение на следването на принципа на наследяването проектът се възползва от възможността лесно да употребява блокове от код повторно и да създава подредена и интуитивна йерархия от класове.
* **Полиморфизъм** – Обектите от различни класове могат да бъдат третирани по един и същ начин, като споделят общ интерфейс.
* **Абстракция** – Скрива се сложността чрез предоставяне на опростен интерфейс за взаимодействие с обекта.

В резултат на прилагането на тези основни принципи, кодът на проекта е организиран и структуриран по начин, който е по-лесен за разбиране, поддържане и разширяване.

## 2.2. Дефиниране на проблеми и сложност на поставената задача

Поставената задача за разработване на SVG parser може да се раздели на подзадачи, които могат да се класифицират като лесни, средно трудни и трудни за разрешаване.

### 2.2.1. Лесни за разрешаване подзадачи

* **Създаване на йерархия от класове, описващи поддържаните фигури** – Тъй като приложението поддържа краен брой от базови SVG фигури, то е дефинирана йерархия от класове, които ги описват. Полетата на тези класове се определят от XML формата на SVG файловете, а методите – от исканите функционалности на приложението.
* **Реализация на методи за принтиране, създаване, изтриване, транслиране и проверка за съдържане в даден регион** – След като са създадени йерархията и съответните колекции от обекти, реализацията на посочените методи е тривиална и се състои най-вече от математически съображения и познаване на функционалностите на достъпните структури от данни.
* **Реализация на методи за затваряне и запазване на файл, даване на информация за функционалностите на програмата и терминиране на програмата** – Състоят се от базови команди за работа с файлове и принтиране в терминала.

### 2.2.2. Средно трудни за разрешаване подзадачи

* **Избиране на ефективна и интуитивна архитектура** – Предизвикателна задача е избирането на подходяща архитектура, която едновременно да спазва основните принципи на обектно-ориентирането програмиране и да бъде интуитивна за използване.
* **Реализация на методите за екзекутиране на програмата** – Трудността на тази подзадача се състои в разнообразието от команди и осъществяването на връзка между множество класове и методи.

### 2.2.3. Трудни за разрешаване подзадачи

* **Отваряне на файл, извличане на информацията и събиране в структура от данни** – Тази подзадача е особено предизвикателна, тъй като изисква особено добро вникване в структурата на SVG файловете. След това са необходими стабилни умения в боравенето с елементарните структури от данни. Налага се дефиниране на допълнителни класове, които отговарят за складиране на специален тип информация и нейната обработка.
* **Прочитане и валидиране на команди и извеждане на подходящи съобщения при въвеждане на невалидни данни** – Трудността на тази подзадача се състои в това, че съществува голямо множество от възможни грешки и некоректности при подаване на информация към програмата. Съответно всяка една трябва да бъде прихваната и за всяка една трябва да се изведе подходящо съобщение.

# Глава 3. Проектиране и архитектура

Вече беше обърнато внимание на това, че проектът е съобразен с принципите и концепциите на обектно-ориентираното програмиране и в тази глава ще вникнем в малко повече детайли относно избраната архитектура и някои основни блокове от кода.

## 3.1. Фигури

Тъй като приложението работи с базови SVG фигури, то имаме създаден базов клас **Figure**, който съдържа полетата, които са общи за всички фигури в SVG формат, както и виртуални методи, които за целта на функционирането трябва да бъдат *override*-нати. Тук прилагаме принципите на наследяването и полиморфизма, като по този начин постигаме целта лесно да може да бъде добавяна поддръжка на нови фигури. До момента приложението поддържа базовите фигури **Line**, **Circle**, **Rect** и съответните класове наследяват базовия клас Figure (фиг.1).

A diagram of a computer

Description automatically generated

фиг. 1

## 3.2. Основни методи за работа с файлове и менажиране на колекции от фигури

За да бъде спазен принципът за енкапсулация, всички методи, които са свързани с работа с файлове и манипулиране на фигурите в тях, са включени в общия клас **FileManager**. Този клас е от тип сингълтън, тъй като не се иска да се създава повече от един екземпляр.

## 3.3. Методи за приемане на команди от командния ред, валидиране и екзекутиране на програмата

За да бъде спазен принципът за енкапсулация, всички методи, които са свързани с четене на команди, валидиране и тяхното екзекутиране, са включени в общия клас **CommandsManager**. Този клас е от тип сингълтън, тъй като не се иска да се създава повече от един екземпляр. Класовете **CommandsManager** и **FileManager** са тясно свързани, тъй като след приемане и одобряване на дадена команда от **CommandsManager**, съответната функция се извиква от инстанция на **FileManager**.

## 3.4. Помощни класове и структури

При разработването на проекта се забелязва многократно използване на блокове от нетривиален код. Поради тази причина се налага дефинирането на няколко допълнителни помощни класове и структури:

* **class Attribute** - описва атрибут на фигура в SVG формат;
* **struct Point** – описва обект от тип точка в двумерна координатна система;
* **class SVGParser** – от тип Singleton, съдържа методи за преобразуване на информация в SVG формат.

# Глава 4. Реализация и тестване

## 4.1. Управление на паметта и алгоритми

Управлението на паметта е от изключителна важност и при използване на динамична памет трябва много да се внимава тя да не изтича. При реализацията на проекта са използвани библиотеките vector, string и cstring, които самостоятелно се грижат за използваната памет. Рискът, който се появява при текущата реализация се появява при създаването на нови фигури. При изпълнението на тази команда фигурата винаги се добавя в основната колекция, която при затваряне на файл или терминиране на програмата специално се изчиства и всяка отделна фигура се изтрива, за да се предотврати изтичането на памет.

В така реализирания проект не са имплементирани специални алгоритми.

## 4.2. Тестове

Съществена част от реализирането на проекта е и неговото тестване. Поради тази причина е създаден специален файл, в който са генерирани автоматични тестове за проверка коректността на изпълнение на повечето от методите.

## 4.3. Използвани помощни инструменти и библиотеки

За разработването на този по-обемен проект са използвани няколко помощни инструмента и библиотеки:

* **ChatGPT** – използван за тълкуване на компилационни грешки;
* **GitHub Copilot –** използван за генерирането на doctest тестове и писане на коментари;
* **Библиотеката vector –** използвана за работа с динамични масиви;
* **Библиотеката cmath –** използване за извършване на по-сложни изчисления;
* **Библиотеките string и cstring –** използвани за работа със символни низове.

# Глава 5. Заключение

## 5.1. Изпълнение на началните цели

Предаденият проект покрива всички зададени задачи. Също така е добавена и нова функционалност – при опит за затваряне на файл или терминиране на програмата при наличие на незапазени във файла промени, приложението допълнително пита потребителя дали иска да запази тези промени и след даване на коректен отговор изпълнява съответната команда.

## 5.2. Бъдещо развитие и усъвършенстване

На този етап предаденият проект покрива всички базови функционалности, но има и редица отворени възможности за бъдещо развитие и усъвършенстване:

* **Добавяне на поддръжка на повече базови SVG фигури** – Дизайнът на приложението позволява лесно това да бъде реализирано. В резултат ще бъдат увеличени възможностите за визуализация на различни изображения.
* **Увеличаване на гъвкавостта на подаваните команди** – Предаденият проект разчита на изключително високо ниво на коректност при подаване на командите от страна на потребителя. Значителна роля за усъвършенстването на потребителското преживяване ще има това валидирането на входните данни да бъде така модифицирано, че в повече случаи приложението да може да изпълнява командите.
* **Подобряване на валидацията** – Както вече е споменато по-горе, множеството от грешни или некоректни входни данни е изключително голямо. Предаденият проект има още много да се развива в посока на това да идентифицира и обработва повече типове възможни грешки на подаваната информация или команда.
* **Добавяне на директна визуализация** – Предаденият проект генерира валидни SVG файлове, които могат да бъдат визуализирани чрез допълнително разширение. В бъдеще би могла да се добави функционалност, която автоматично да визуализира текущо правените промени по файла.

# Примерно използване на приложението:

> close

No file loaded!

> open test.svg

Successfully opened test.svg

> print

(1) circle 15 25 5 green

(2) rectangle 11 15 10 2 blue

> translate

Expected more arguments!

> translate vertical=5 horizontal=6

Translated all figures!

> save

Successfully saved file test.svg

> print

(1) circle 21 30 5 green

(2) rectangle 17 20 10 2 blue

> erase 3

There is no figure number 3!

> erase 1

Successfully erased rectangle (1)!

> close

There are unsaved changes. Do you want to save them? (yes/no)

> no

Successfully closed file test.svg

> open fig.svg

New file created.

> print

No figures in current file!

> create circle 0 0 5 blue

Successfully created circle (1)!

> save

Successfully saved file fig.svg

> within circle 0 0 6

circle 0 0 5 blue

> open test.svg

File already loaded!

> erase 1

Successfully erased circle (1)!

> save

Successfully saved file fig.svg

> print

No figures in current file!

> cloase

Invalid command. See help for more information!

> close

Successfully closed file fig.svg

> open test.svg

Successfully opened test.svg

> print

(1) circle 21 30 5 green

(2) rectangle 17 20 10 2 blue

> create rectangle 5 4 8 8 green

Successfully created rectangle (3)!

> saveas figures.svg

Successfully saved file figures.svg

> open figures.svg

File already loaded!

> close

Successfully closed file test.svg

> open figures.svg

Successfully opened figures.svg

> print

(1) circle 21 30 5 green

(2) rectangle 17 20 10 2 blue

(3) rectangle 5 4 8 8 green

> translate 3 vertical=5 horizontal=10

Translated figure (3)!

> exit

There are unsaved changes. Do you want to save them? (yes/no)

> yes

Successfully saved file figures.svg

Exiting...

# GitHub Repository

През цялото време на разработване на представяния проект текущите промени са качвани в Repository на следния линк: <https://github.com/DanaBachvarova/SVG-files>