**Работа с SVG файлове**

**Технически университет – Варна**

Факултет: ФИТА

Катедра: Софтуерни и интернет технологии (СИТ)

Специалност: **СИТ**

Изготвил: **Тодор Тодоров Павленков**

Факултетен номер: **21621536**

**Структура на документацията**

1. Увод

a. Задание

2. Преглед на предметната област

a. Основни дефиниции, концепции и алгоритми

b. Дефиниране на проблем и сложност на поставената задача

c. Подходи, методи за решаване на поставените проблеми

3. Проектиране

a. Обща структура на проекта

b. Блок схема

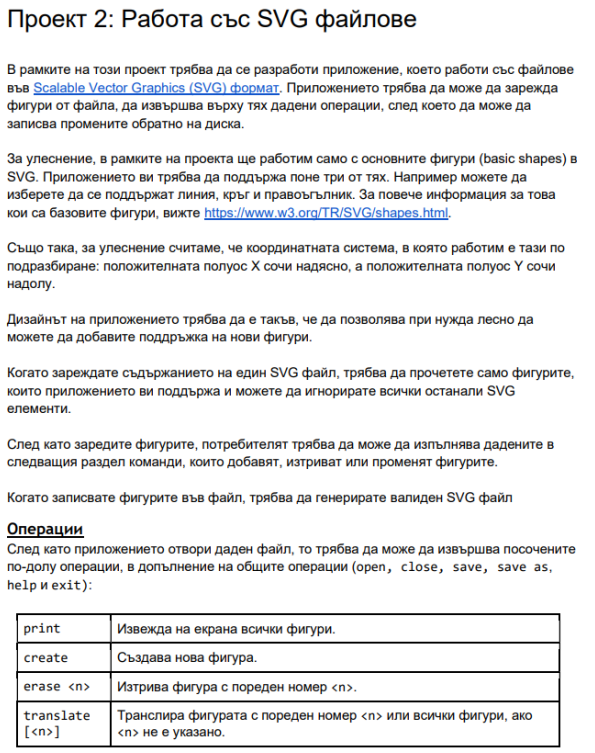
4. Реализация и тестване

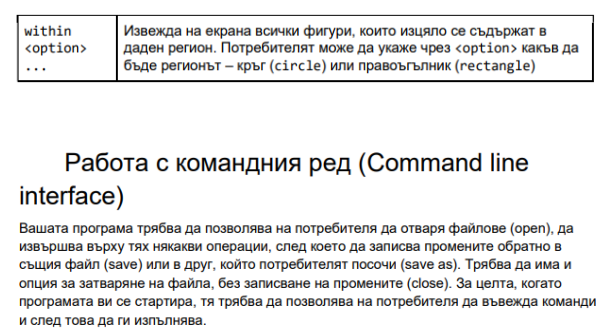
a. Реализация на класове

b. Планиране, описание на тестови сценарии

1. Увод

a. Задание





2.Преглед на предметната област

a. Основни дефиниции, концепции и алгоритми ***SVG (Scalable Vector Graphic)*** – Мащабируема векторна графика е тип формат **базиран на** **xml**, който има за цел да създава графини изображения с помощта на вектори

b. Дефиниране на проблем и сложност на поставената задача

***Проблем 1***: Да се реализира програма, която о работи със файлове във ***Scalable Vector Graphics (SVG)*** формат. Приложението трябва да може да зарежда фигури от файла, да извършва върху тях дадени операции, след което да може да записва промените обратно на диска.

***Проблем 2***: Дизайнът на приложението трябва да е такъв, че да позволява при нужда лесно да може да се добавя поддръжка на нови фигури. Когато се зарежда съдържанието на един SVG файл, трябва да се прочетът само фигурите, които приложението поддържа

Сложност на поставената задача:

Високо ниво на абстрактно и логическо мислене за реализиране на програмата.

c. Подходи, методи за решаване на поставените проблеми

**Решаване** на ***проблем 1***: При отварянето на даден файл се зареждат всички фигури под главния “” таг, след това поддържаните фигури се филтрират посредством функцията ***isShapeSupported***, която сравнява тагът на текущата фигура с стойностите на енумерацията ***SupportedShapes***.

**Решаване** на **проблем 2**: Добавянето на нова фигура става сравнително лесно, като първо се добавя нова стойност в енумрецията с поддържани фигури SupportedShapes, като там ще се съдържа информация за тагът на съответната фигура броят на параметрите и.

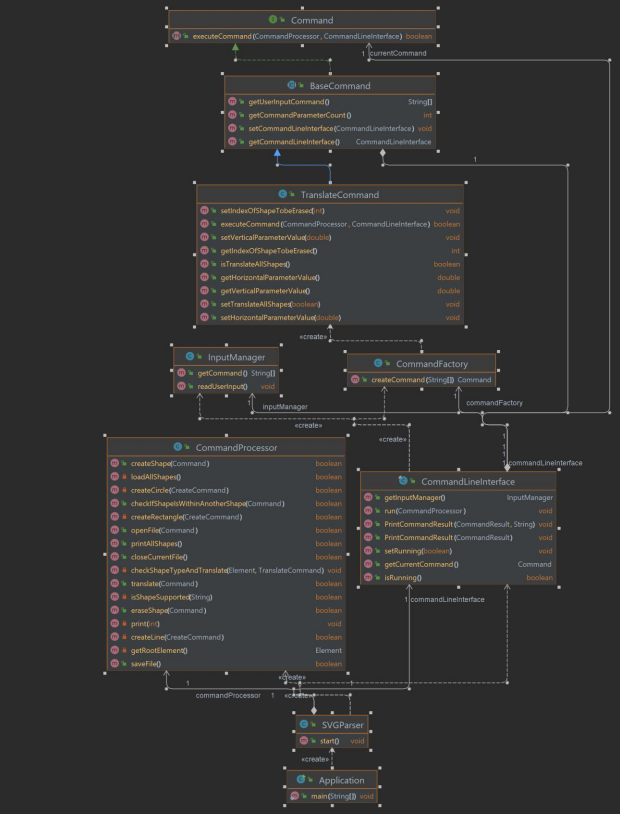
Втората стъпка е да се добави нов пасаж за новата фигура във валидиращият клас(***CreateCommandValidator***) който ще направи проверки дали въведените параметри са коректни и дали фигурата се поддържа.

3. Проектиране

a. Обща структура на проекта

1. Kлас ***CommandProcessor*** съдържа логиката нужна за изпълнение на командите
2. Клас ***InputManager*** служи за четене на входни данни от потребителя
3. Клас ***CommandFactory*** служи за създаване на командите
4. Интерфейс ***Command*** дава възможност за създаване на имплементации на различните комадни
5. Интерфейс ***CommandValidator*** дава възможност за създаване на имплементации на различните валидиращи класове
6. Клас ***CommandLineInterface*** – съдържа композиция от класовете InputManager и CommandFactory служи за обработка на входни данни и създаване на командите
7. Съответно класовете ***OpenCommand***, ***CloseCommand***, ***SaveCommand***, ***SaveAsCommand*** и тн. Представляват имплементации на поддържаните команди от приложението, като целта е с помощта на валидиращите класове всяка команда само да се определя и да се валидира.

b. Блок схема

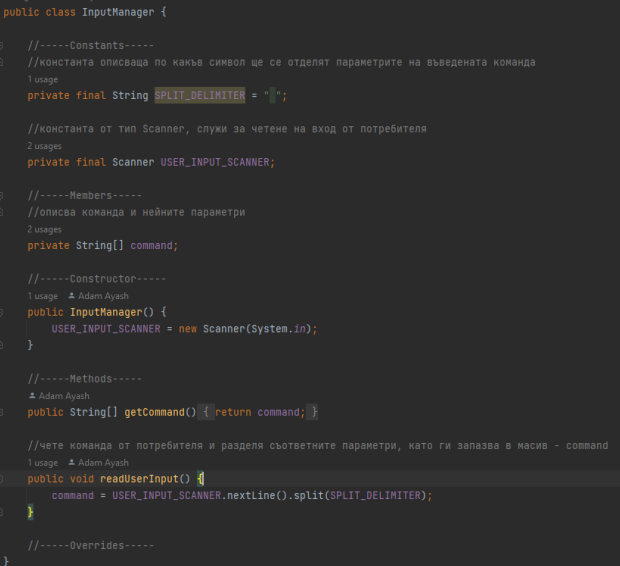


4. Реализация и тестване

a. Реализация на класове

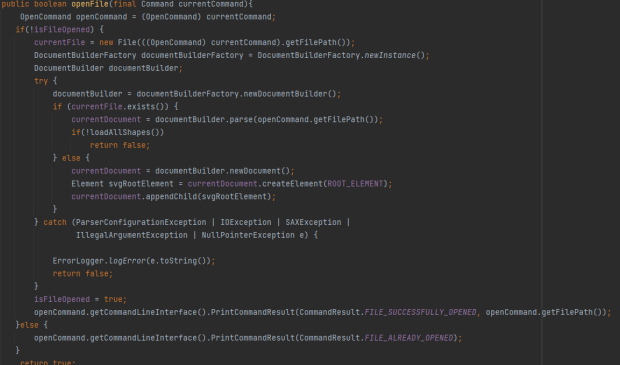
Четенето на входни данни от потребителя става посредством ***InputManager*** класът, който след прочитането на съответната команда и нейните параметри я разделя по интервал и запазва всеки елемент в масив от низове.

Фигура 1:



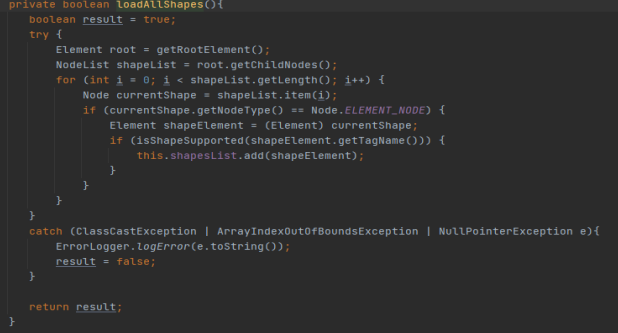
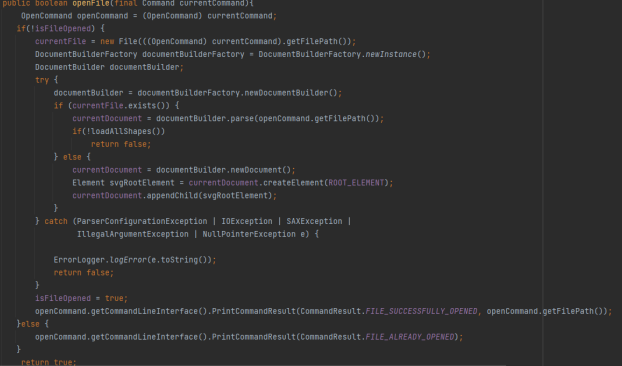
Създаването на командите става посредством класът ***CommandFactory*** и методът **createCommand** фигура 2 , който проверява дали командата е поддържана и ако това условие е изпълнено я създава.

Фигура 2:



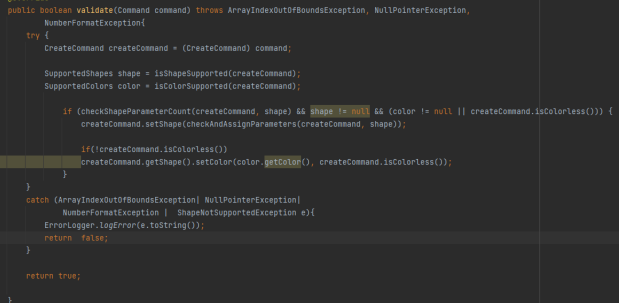
**Отварянето** на **файловете** и **зареждането** на **фигурите** става чрез класа ***CommandProcessor*** и методите ***openFile*** фигура 3 и методът ***loadAllShapes*** фигура 4.

Фигура 3:



Създаването на фигури става посредством класът ***CreateCommand***, който има за цел да създаде транспортен клас за атрибутите на фигурата, като за да се определи дали параметрите на фигурата въведени от потребителя са коректни командата минава през валидиращ клас ***CreateCommandValidator*** метод **validate** фигура 4.

Фигура 4:



d. Планиране, описание на тестови сценарии

При стартиране на програмата, първоначално се изисква да се отвори ***xml*** файл. Фигура 5

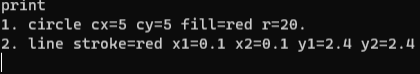
Ако се напише друга команда се извежда съобщение, че трябва да се отвори файл.

Фигура 5:



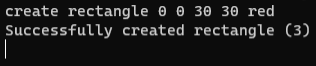
Файлът ***file.svg*** съдържа предварително въведени фигури. С команда ***print*** можем да ги изведем на екрана .

Фигура 6:



За създаване на нова фигура използваме команда ***create***.

Фигура 7:



За манипулация на координатите на фигурите използваме командата ***translate***

Фигура 8:



За проверка дали наличните фигури се съдържат в зададена от нас фигура използваме командата ***within***

Фигура 9:

