Документация на курсов проект на тема „Работа със SVG файлове“

Съдържание на документацията:

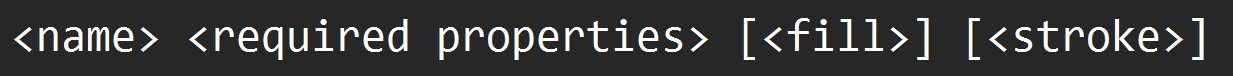
1. Увод
   1. Идея на проекта
   2. Цел при разработката на проекта
2. Структура на кода и настъпили усложнения
3. Архитектура на класовете
4. Тестване
5. Възможности за подобрение и разрастване
6. [Линк към хранилището в *github*](https://github.com/Victor147/OOP-Final-Project)
7. Увод

Документацията към курсовия проект „Работа със SVG файлове“ има за цел да представи основните стъпки при създаването на приложението, архитектурата на класовете, решенията на казусите, възникнали по време на разработката, направените тестове.

Основната задача на проекта е да се създаде „редактор“ на файлове, съдържащи векторно изобразени елементи. Това е най-олекотената форма на редактира на SVG файлове – прочитане на съдържанието на съществуващ файл, създаване на нов векторен обект, преместване на някой или на всички елементи на файла, премахване на фигура и команда, чрез която е възможно да се разбере колко от фигурите се намират в зададен кръг или правоъгълник. Програмата работи с 6 от 7 възможни „прости фигури“, достъпни в SVG файловете. Едно от ключовите изисквания към проекта е да бъде представен чрез подходяща архитектура на класовете и да позволява лесно добавяне на нов елемент към йерархията от класове. Добавянето на функционалност за нова фигура може да се осъществи сравнително лесно – като се добави нов наследник на базовия клас.

1. Структура на кода и настъпили усложнения

Основната по-сложна задача в проекта е тази за извличането на информация от SVG файл. Понеже всяка от фигурите има различни атрибути, които я представят, се усложняват нещата с текстообработката. Освен това при създаването на фигура се налага са бъде въведена структура, която се разпознава от програмата, защото в противен случай могат да бъдат подадени стойности, които отговарят на други атрибути, които не се поддържат. При прочитането от файл се запазват всички подадени свойства. На фигурата се вижда възможния формат за подаване на данни при създаване от конзолата.



Атрибутите *fill* и *stroke* не са задължителни, но могат да бъдат подадени, за да има по-пълно описание обекта. Всички параметри, които преминават след тази граница не биват запазени в съответния обект.

Основният принцип при създаването на обекти от базовия клас Figure става чрез *Factory Pattern*-a. Този шаблон не се различава от своето типично имплементиране, като е добавен и нов параметър към създаващата функция – вектор със стойностите на основните атрибути на шестте фигури и евентуално допълните особености на всеки отделен елемент. Съхраняването на тези създадени указатели се съхранява в поле на класa *SVGFile,* което е хетерогенен контейнер. Тук е имплементиран един от основните принципи – полиморфизма. През класа *SVGFile* се извършват всички действия, които са пряко свързани с всеки един от обектите в контейнера. Този клас играе ролята на междинен клас, защото свързва всяко едно конкретно действие към вектора с указателите и по този начин е възможно изпълнението на всяка методи в класовете-наследници на *Figure*.

1. Архитектура на класовете

A diagram of a software system

Description automatically generated with medium confidence

Класът *Menu* се използва за обработка на входните команди от конзолата. Там се осъществява отсяването на валидните и невалидните команди, като също се създава и обект от класа *SVGFile,* върху който се извършват операциите, заложени в заданието на проекта. Полетата на класа *SVGFile* са няколко – низ за името на файла и низ за местоположението му във файловата система, които се използват при командата *save* и колекцията от всички фигури. Последният съществен компонент е този на базовия клас и неговите наследници, които имплементират следните методи: *print, translate, contains, within, readFromFile, save, getType.* Десериализирането на данните става посредством виртуалната функция *readFromFile,* която се имплементира от всеки наследник и по този начин може да се дефинира операторът (>>). Обратният процес на сериализацията се извършва от метода save.

Има налични и два помощни класа – *Point* и *Property*, които спомагат за по-лесната работа с наличната информация за обектите. Класът *Point* предлага представяне на точка от геометричната равнина, като са налични методи за преместване на точка, намиране на разстояние до друга точка, както и селектори и мутатори. Класът *Property* служи за обработката на всички атрибути на фигурите, които не са задължителни за съществуването им. Обекти от този клас могат да бъдат – цвят на фигурата, цвят на контура и други. При тях информацията се съхранява в стринг. Този клас освен стандартните методи предлага и метод за добавяне на информация към конкретен атрибут – той е много полезен при началното запазване на информацията от подаден файл.

1. Тестване

Налични са три тестови файла, както и примерни команди към тях. Чрез тези файлове може да се извърши нагледна проверка на всички възможности, предлагани от програма – наличните команди, невалидни входни данни и т.н. Освен тези тестови случаи са налични и *unit* тестове. За *unit* тестване е използвана библиотеката *doctest,* която предоставя един заглавен файл, който се добавя към проекта. За изпълнение на тестовете не се изисква никаква сложност, а само да се добави заглавен файл, и след това да се изпълнят малко на брой команди в *main* функцията.

A screen shot of a computer code

Description automatically generated

Основната цел на тестването е да се провери изправността на всеки компонент, независимо от другите. Това на практика е почти невъзможно да се осъществи, затова стремежа в тестовия код е постепенно да се премина от по-независими компоненти към такива, които са по-обременени. Тестовете са основно базирани към методите на класовете, свързани с *Figure* и към класа *SVGFile.* Това е така, защото там се намират основната логика на приложението и всичките по-важни елементи.

1. Възможности за подобрение и развитие
   1. Добавяне на функционалност за намиране на лица, дължини (периметри) – за тази функционалност има добавени методи за намиране на лицето на кръга, правоъгълника и елипсата, за техните периметри и за дължината на линията и *polyline*-а.
   2. Може да се добави групиране на елементи, като по този начин приложението да се доближи още повече към истински редактор.
   3. Добавяне и на най-сложната от всички базови фигури – *path*
   4. Оптимизиране на структурата на кода, както и на алгоритмите
   5. Добавяне на функционалност за ротация и транслация на телата