**Parsing, working and generating valid SVG files:**

This is my final project at Object-oriented-programming-OOP university course. I must admit that the project is not fully completed. The unfinished part of the project is mostly connected with files. The whole work with polymorphism is finished and implemented. That's a console application which is the final project of my Object oriented programming university course in St. Kliment Ohridski university of Sofia. It is finished by the date of: 05.07.2016.

**The way your program shoud work**

Your program should give the user the ability to open files (open), to do some operations upon them and after that to save the changes back on top of the same file (save) or on top of another file which the user will specify (save as). There should be an option for closing the file without saving the changes (close). For that purpose, after the program is started the user should be able to input different commands and execute them.

When you open certain file it's content has to be loaded in the memory and after that the file should is closed. All changes which the user does after that should be kept in the memory, but they shouldn't be saved on top of the file content unless the user dont give a command for it.

Your program should:

1. be able to read a random file from the file format which is specified below;
2. be able when saving the changes to create again a valid file from the same format.

As we said above the user must be able to input commands whereby to specify what exactly to be done. The commands shoud have zero, one or more parameters, which are given one after another separated with spaces.

If it's not specified any command must display a message which shows if the operations is executed successfully or not.

The commands must be supported from every project (below every command there's an example):

* Open: loads the content of a file. If the specified file doesnt exist, it creates new one with an empty content. The rest of the commands shoud be executed only if there's successfully loaded file.

example:

open C:\Temp\file.xml Successfully opened file.xml

* Close: close the file.

example:

close Successfully closed file.xml

* Save: save the changes on top of the same file from which the content is taken.

example:

save Successfully saved file.xml

* Save As: save the changes in a file, while giving the user the ability to specify its path.

example:

saveas "C:\Temp\another file.xml" successfully saved another file.xml

* Exit: exits the program.

example:

exit

**Project 1: SVG Files.**

You should create a console application which works with SVG files. The application must be able to load figures from the file, to do on them given operations and after that to save all changes.

For ease the project should support only the basic shapes in SVG. Your application shoud support at least three of them. For example you coud choose to support line, circle and rectangle. For more information take a look at: <https://www.w3.org/TR/SVG/shapes.html>.

Also, for ease we consider that the coordinate system, upon which the program works, is the same as the default one or the X coordinate pointing right and Y coordinate pointing down.

The design of the application should be maintainable and new functionality shoud be easily to adopt.

While you are loading the content of an SVG file you can only read the shapes your program supports the rest of them SVG elements you can ignore.

After you load the file, the user should be able to execute the commands specified below who adds, deletes or changes figures.

When you save the figures in a file you should create valid SVG file.

**Operations executed upon the file content:**

*Print* prints all figures.

*Create* creates new figure.

*Erase* deletes figure.

*Translate* Translate one or all shapes. If the user don't specify certain shape all shapes will be translated. If the user specify a shape only that figure is translated.

*Within* prints all figures, which are in specific region. The user can specify what region, area will be a circle or rectangle.

**An example of SVG file figures.svg:**

**An example how your application should work:**

*open figures.svg*

Successsfully opened figures.svg

*print*

1. rectangle 5 5 10 10 green

2. circle 5 5 10 blue

3. rectangle 100 60 10 10 red

*create rectangle -1000 -1000 10 20 yellow*

Successfully created rectangle (4)

*print*

1. rectangle 1 1 10 20 green

2. circle 5 5 10 blue

3. rectangle 100 60 10 10 red

4. rectangle 1000 1000 10 20 yellow

*within rectangle 0 0 30 30*

1. rectangle 5 5 10 10 green

2. circle 5 5 10 blue

*within circle 0 0 5*

figures are located within circle 0 0 5

*erase 2*

Erased a circle (2)

*erase 100*

There is no figure number 100!

*print*

1. rectangle 5 5 10 10 green

2. rectangle 100 60 10 10 red

3. rectangle 1000 1000 10 20 yellow

*translate vertical=10 horizontal=100*

Translated all figures

*print*

1. rectangle 105 15 10 10 green

2. rectangle 200 70 10 10 red

3. rectangle 1100 1010 10 20 yellow

*save*

Successfully saved the changes to figures.svg

*exit*

Exit