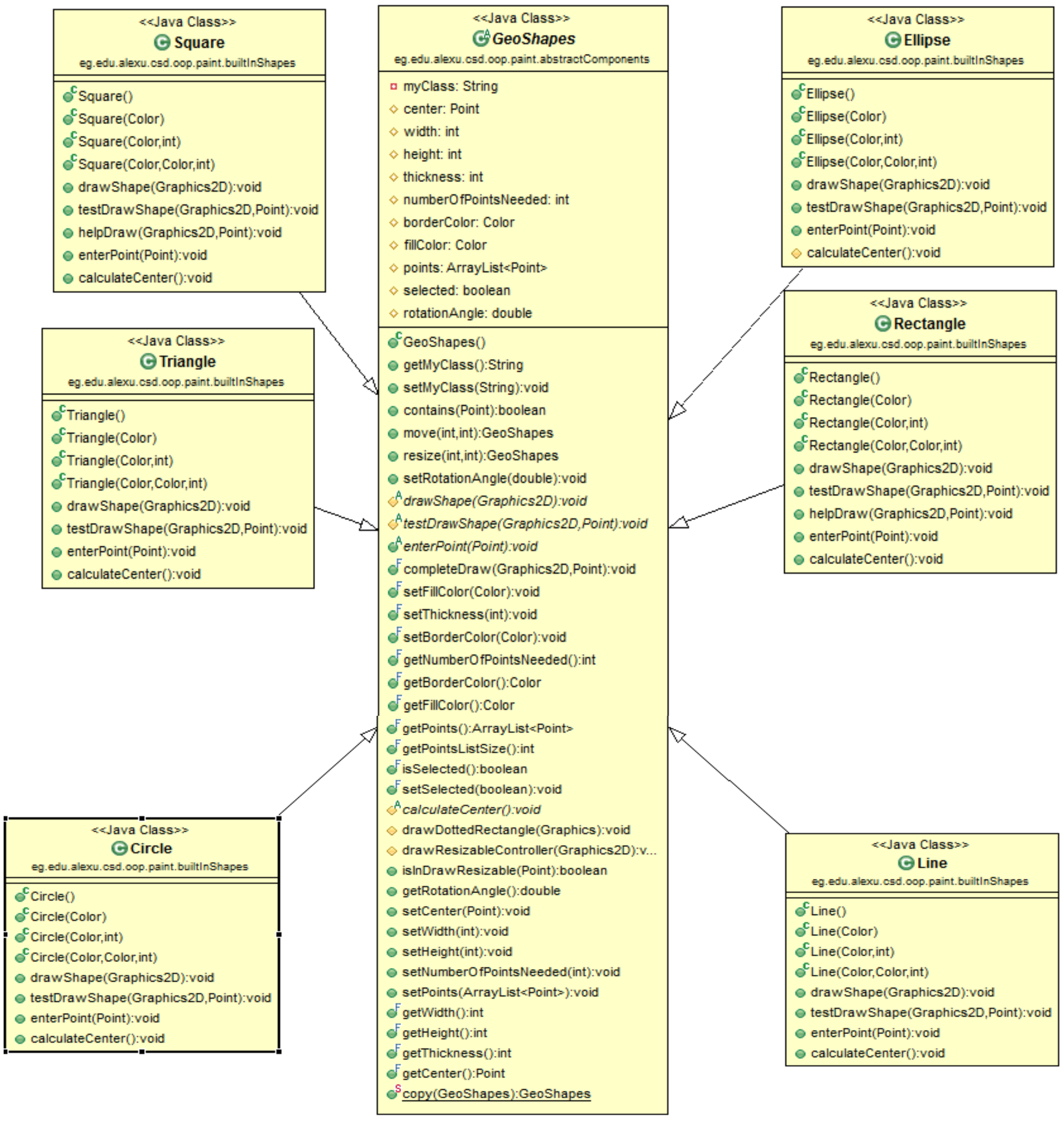
CSED 19 - OOP

Paint Report

By Amr Mohamed Nasr & Michael Raafat Mikhail

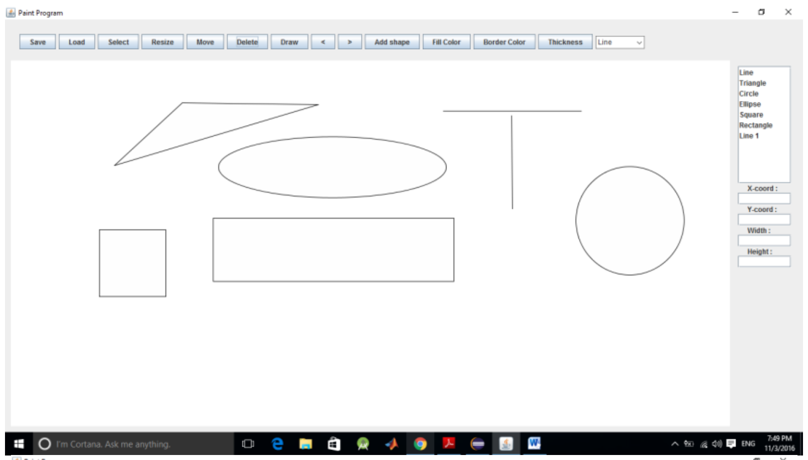
Reqired UML diagram

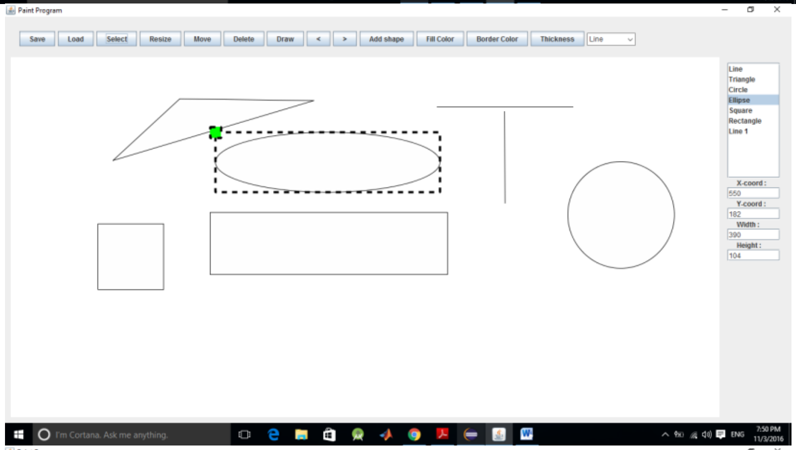


User guide

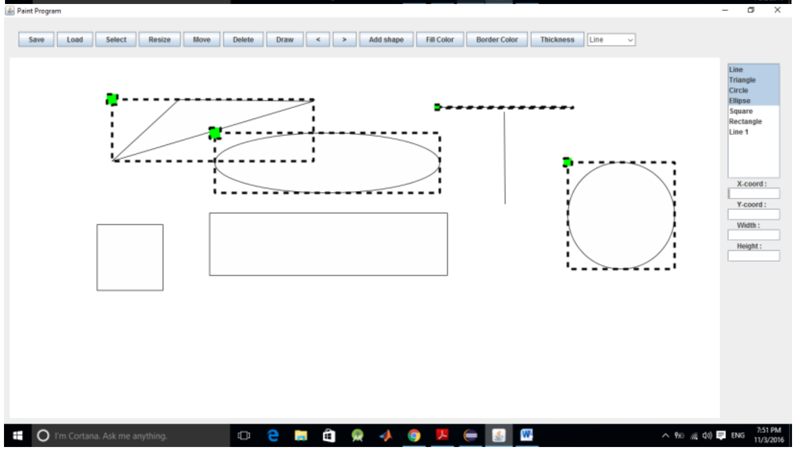
1. Our paint program enables you to draw any geometric shapes you desire and color their borders and backgrounds with colors you prefer.
2. At the center, you can find your drawing area.
3. A guide for program's buttons :
    1) Save: to save your final work in XML or JSON file.
    2) Load: to load the work you saved.
    3) Select: to choose the shapes you want (you can choose more than one)
        - To deselect a shape: click on it.
    4) Resize: to resize the shapes you select.
    5) Move: to move the shape to where you want.
    6) Delete: to delete shapes you don't want.
    7) Draw: to enable you to draw the shape you desire from the list at the upper right corner.
        - You click to set your shape's first point, then you move your mouse to draw, and when you want to set you next point just click.
    8) Redo ">" and Undo "<": to undo or redo the last action you have done.
    9) Add shape: to add a new shape you want to the list to use it later.
    10) Fill color: to change the color of shape's background.
    11) Border color: to change the color of the border.
    12) Thickness: to change the thickness of border.
4. A list on the right side of our program where you can select the shape and adjust its width, height and co-ordinates.

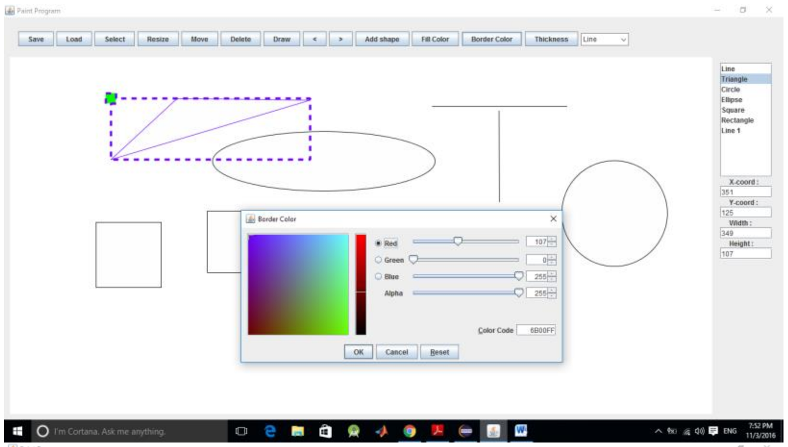
<u>Snapshots of our GUI</u>





X-coord : 550
Y-coord : 182
Width : 390
Height : 104

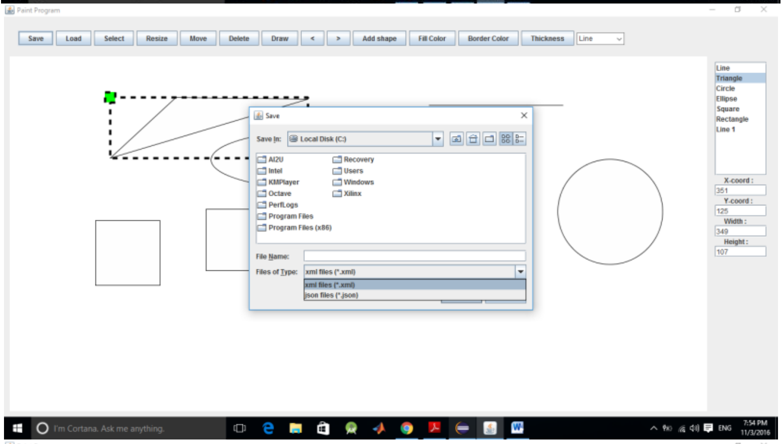




X-coord : 351
Y-coord : 125
Width : 349
Height : 107

Red 107
Green 0
Blue 255
Alpha 255
Color Code 6B00FF

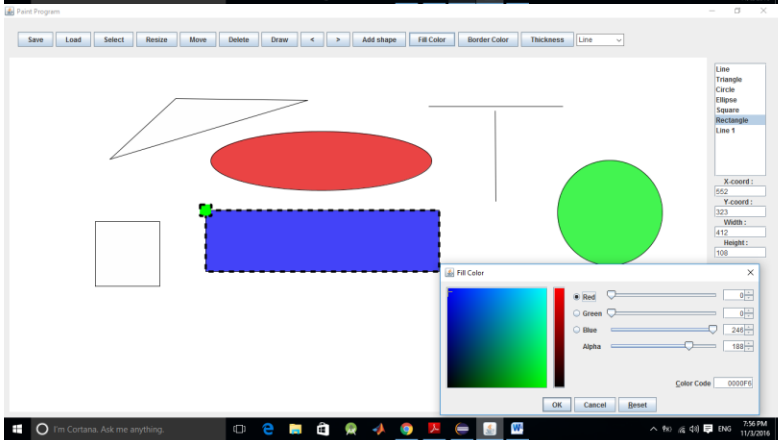
7:52 PM 11/3/2016



X-coord : 351
Y-coord : 125
Width : 349
Height : 107

Save In: Local Disk (C:)
File Name:
Files of Type: xml files (*.xml)
xml files (*.xml)
json files (*.json)

7:54 PM 11/3/2016



X-coord : 552
Y-coord : 323
Width : 412
Height : 108

Red 0
Green 0
Blue 246
Alpha 188
Color Code 0000F6

7:56 PM 11/3/2016

<u>Our design</u>

1. Abstract Components:
    a. GeoShapes:
        Abstract class containing the blue print of all shapes and allows the implementation of any shape the user wants. It contains the implementation of some basic functions and will ask the user to implement his own. This class will be what the other components deal with. Not the classes that extend it.
2. Built in Shapes:
    Classes that extend the abstract class GeoShapes and will be considered available on the loading of the program.
    a. Circle
    b. Triangle
    c. Rectangle
    d. Ellipse
    e. Square
    f. Line
3. Interfaces:
    The interfaces for the main building blocks of the functionalities and those that can be implemented differently and we chose to offer the option to change the implementation as the user wants without having to change other things in the program.
    a. IDrawingDataCore
        This is the interface for the object that should hold all the information needed of the shapes, of the user actions and attributes of the drawing. Responsible of the operations that change attributes with some primary input not needing action listeners.
        Dependences: commandHandler classes as the controlling unit of history.

b. IDrawingMouseReaction

   This is the Interface responsible of the actions taken in case of mount events. Meaning it will be responsible of the advanced operations and functionalities of the drawing with anything that the user mouse will be responsible of and any advanced function.
   Dependences: DrawingBoard class.

c. IPaintSaver

   Interface of the class responsible for the saving, loading functionalities. Will save or load an object that implements the IDrawingDataCore interface.

4. Helpers
   a. Config

      A utility class that is responsible of giving the different constants used to the classes. Offers a job that is like an options pane where we give in the config the implementation class of each interface and the default values for different settings in our project. Meaning that to make a change you don't have to change in each class, it will be enough to only change in this file. Almost any class will depend on Config.
5. File Manager
   a. WriterReadFile

      Class that implements IPaintSaver.
      Depends on : xStream Library, Jettison Library, IDrawingDataCore.
6. Command Handler
   a. ShapeNode

      A class that represents a node containing an index and a GeoShape.
   b. InfoCapsule

      A class that contains all the information needed to recreate an action made by the user or to reverse it.
      Depends on: ShapeNode.

   c. CommandManager

      The class responsible of keeping the history and saving it. It is also responsible of undoing or redoing.

      Depends on: ShapeNode, InfoCapsule, IDrawingDataCore.

7. Draw Components

   a. DrawingMouseAdapter

      A custom mouse adapter class that uses an IDrawingMouseReaction implementation class to define the actions done in each event.

   b. DrawingPen

      An implementation class of the IDrawingMouseReaction.

      Depends on: IDrawingDataCore, Command Handler classes.

   c. DrawingData

      An implementation class of the IDrawingDataCore.

      Depends on: Command Handler Classes.

   d. DrawingInspector

      A Gui object that extends JPanel, responsible of linking into the used data core of the painting. Keeping a hierarchy of the shapes up to date and getting the properties of selected shapes. Also, offers the power to select things more precisely and offers more precise control on the properties of selected shapes.

      Depends on: IDrawingDataCore, GeoShapes.

   e. DrawingBoard

      The main GUI component. It extends JPanel and will be, as its name inspires, the board where the shapes will be drawn on. It is the component responsible of synchronizing all the other drawing components and will encapsulate all of them. Meaning that to call a function from any of them will only be available by calling it from a provided function in the board.

      Depends on: IDrawingDataCore, IDrawingDataMouseReaction, DrawingInspector, DrawingMouseAdapter.

8. Paint
    a. PaintGUI

       The main class for our project. Responsible of the GUI functionalities and will offer the processing of most of the user input. Will also contain the functionality of the dynamic class loading and will encapsulate all the other functionalities of the different components.

       Depends on: IPaintSaver, DrawingBoard.