Computer Programming A, B, E
FAST-NU, Lahore, Spring 2017

Homework 5

Paint Brush using Inheritance and Graphics

Due on Saturday April 22 11:55 P.M. Marked out of 100 points.

**Objectives**

The objectives of the project are twofold:

1) To implement the Object Oriented Programming concepts learnt in class, such as Inheritance and Polymorphism.
2) To work with a graphics library, and learn how to write an event driven program.

# SECTION 1: Application Features

Our application is a basic drawing utility like the Paint Brush software on Windows. This program will allow the user to draw basic shapes such as triangles, rectangle, polygons circles, lines, etc. The user chooses a particular color from a palette to draw these shapes, and also can use a bucket-tool to fill up the interiors of closed shapes, an eraser tool to delete a shape, and save and load from a file on the disk.

- The drawing mechanism will work as follows:

  - The user will first choose the type of shape he wants to draw from a panel of available shapes.
  - The user will then click the points, that is, the vertices of the shape, and then the program will complete the shape using these points. For example, if the user choose the triangle shapes and then left-clicks the mouse on three different points on the canvas, those three points will be used to draw the triangle.
  - Once a shape has been drawn, it will be added to a list (an array of shape type pointers) of all shapes, called *allShapes*, where the objects of the various shapes drawn so far have been dynamically allocated and their pointers (of type shape) has been stored.
  - Note that the array allShapes itself needs to be dynamically allocated, as its size might vary as new shapes are added or deleted. At any point its size should be **no more than** twice the number of shapes stored in it. (It's size with size=1, and each time it runs out of space the size is doubled).

- Shapes can be deleted as follows:

  - The user will pick the "eraser" tool from the panel.
  - The user can then click anywhere on the canvas, and if the point of the click is inside a shape, that shape will be deleted from the allShapes array.
  - This will go on as until the user un-picks the eraser tool.

- Shapes can be filled as follows:
  **Note:** this tool will only change the boundary color in case of curves and lines.

  - The user will pick the "bucket" tool from the panel.
  - The user can then click anywhere on the canvas, and if the point of the click is inside a shape, that shape will be filled with the current color.
  - This will go on as until the user un-picks the eraser tool.

- The colors: the program will allow (at least) Red, Green, Blue, Yellow, Orange, White, Black
- The file can be saved as follows:

  - At any point during drawing the user can select the save option from the panel.
  - The program will then save all the shapes in the allShapes array in a file called "drawing.pb".

- A file can be loaded into the program at the start:

  - When the application starts it will ask the user to either load the saved drawing or start a new one.
  - If he chooses to start a new drawing the older one will be overwritten, otherwise the contents of drawing.pb will be loaded into the allShapes array and the user will proceed drawing from there.

- What will the panel look like:

  - The panel is supposed to be a vertical stripe on the left side of the canvas with square shaped "buttons", clicking which the user can select various options.
  - There are buttons for: shapes, colors, eraser tool, bucket tool and save option.

# SECTION 2 Classes and Hierarchy

The array **allShapes** contains Shape type pointers to the various objects that have been drawn so far. All these objects are of classes belonging to the Shape hierarchy, with an abstract class called Shape on the top. Here's a point-wise description of this hierarchy.

- In the following "**Point**" is a struct containing "doubles" x and y, for the x and y coordinates of a point on the canvas.

**THE CLASSES**

- What are the various shape allowed by the application:

  - At minimum the program allows the user to draw the following (the number of Points the shapes contain are given in the brackets after each)

  - **Line** (2: the endpoints), available is two stiles: dotted and solid.
  - **Triangle** (3: the three corners)
  - **Rectangle** (2: two opposite corner)
  - **Circle** (1: the center, although the user will have to specify the radius by clicking a second time on any point on the required circumference)
  - **Polygon** (n: any number of points the user wants to specify, he will eventually click the right mouse button to indicated that he has given all the points)
  - **Curve** (**n:** any number of points the user wants to specify, he will eventually click the right mouse button to indicate that he has given all the points.)
    - **Note:** for us a curve is drawn similarly to a polygon (by joining points with straight lines, but it is not closed like a polygon: first and last points are not joined back)
  - **Text (1:** the coordinates of the point on screen where the text starts)

**THE HIERARCHY**

- You should infer the Hierarchy from the following statements:

  - A Shape is either an OpenShape, a Circle, a Polygon, or Text

- An OpenShape is either a line or a curve.
- A Polygon: could be a general Polygon (with n points), or a Rectangle (with just two points: for two opposite corners)

➢ Attributes common to all Shapes
- A list (array) of Points, that are vertices of that shape.
- Integer, called color: indicating the boundary color (or text color in case of text)
- const Integer, n: number of points (length of string in case of text)

➢ Attribute common to all Polygons

- Integer, fillColor: indicates the color with which the polygon is filled. If it is not filled this attribute contains some exceptional value.

➢ Attribute special to a Circle:
- Integer, fillColor: indicates the color with which the circle is filled. If it is not filled this attribute contains some exceptional value.
- Double, radius: radius of the circle

➢ Attribute common to all OpenShapes:
- Boolean, style: true if the line is solid, false otherwise.

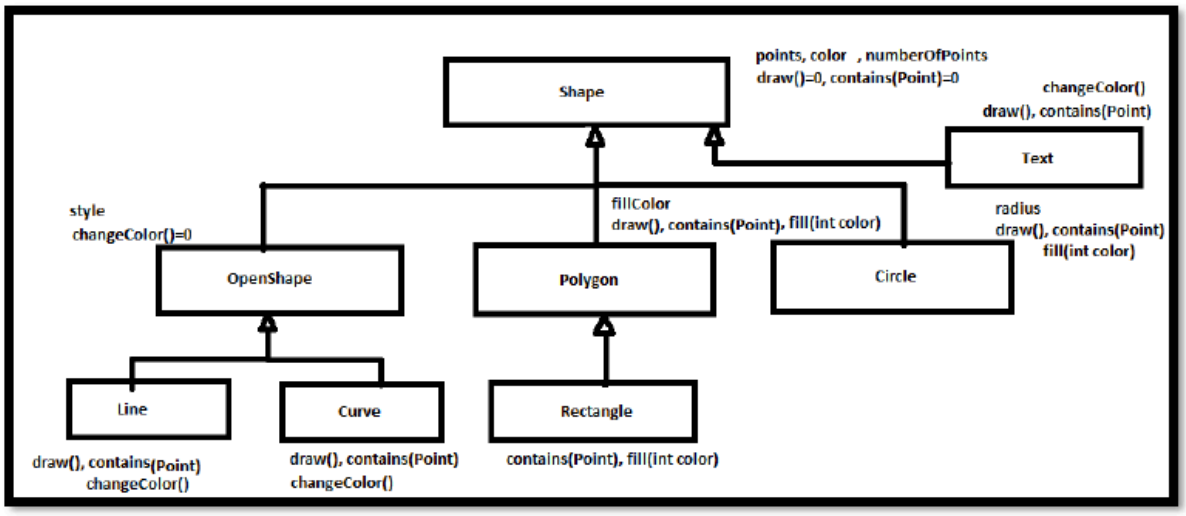
➢ Methods common to all Shapes
- **void draw():** draws the shape on the canvas, and immediately stores the new shape in the allShapes array. (In case of text the letters are already printed as they were typed, so draw will simply create a Text type class and add to all shapes).
- **contains(Point p)**: returns true is the shape contains Point p either in the interior or on the boundary. Lines and curves will also implement this function and return true if p lies anywhere "on" them.
  - To implement this, read about the **Ray Casting Algorithm**: which is a very simple way to detect if a point is inside or outside a closed shape.
  - In case of lines and curves you simply need to detect if the point is on the edges making them up.
  - You will find it useful to implement a utility function called pointLiesOnLine(Point a, Point b, Point c), where the first two points are endpoints of a line and the third point is to be tested: if it lies on the line you return true, else false.
  - In case of Text contains will return true if the click was anywhere on the letters of a text.

- Methods common to all Polygons, and the Circle
  - **void fill(int color)**: fills the closed shape with the given color
    - Repeated use of contains function will help you do this.

- Method common to all OpenShapes
  - **void changeColor(int color)**: changes the boundary color.

- Method special to Text
  - **void changeColor(int color)**: changes the color of the text.

The Hierarchy is summarized in the following picture. Next to each class I have mentioned the attribute(s) that it needs to have, and the function that it needs to either define as pure virtual (in case of Shape and OpenShape which are both Abstract Classes); or override, in cases where the function needs to be specialized for the child classes.




**The Shape Hierarchy**

# Section 3 Graphics Library

You must use GP142 as taught in lab.

The only library functions allowed to be used are line and circle… all other functions you must create using these two methods.

# SECTION 4

**Additional Points on Coding**

- While loading from the file the program will allocate all shape objects for the saved shapes and put them in the allShapes array. Then it will go throw the array and draw (and fill in required) each shape on the canvas.
- There should be no memory leaks anywhere in the program, at any time during execution.
- When the eraser tool is used to delete a shape the pointers will have to be shifted back by one place each, and if the number of remaining shapes becomes less than half the size of the allShapes array then the size of the array should be halved. This will allow us to maintain the condition that the size of allShapes should never be more than double the number do shapes it actually stores.

- **For further questions, post on the Google group.**

THE END