Programming II

Painter Project

Prepared by:

Name
Arsany Moussa Fathy Rezk
Youssef Samuel Nachaat Labib
Youssef Amr Ismail Othman

Classes Description

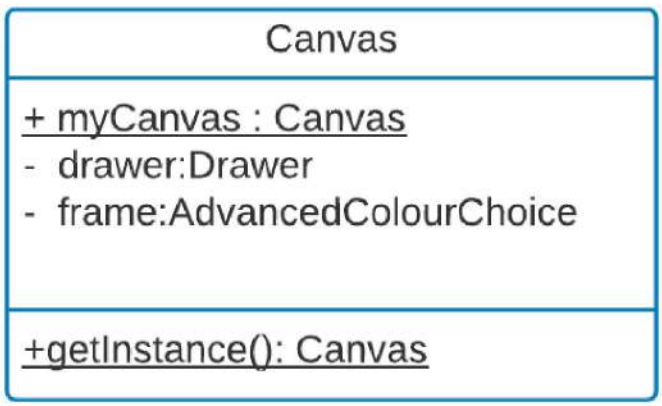
1. Canvas Class

This is our GUI Class where the user is free to do all the available processes we provide in the programme as (Draw shape, Resize shape, Delete shape, etc…).

Once an instance of Canvas Class is created in the driver class (Painter) a drawing board and many buttons are to be displayed to the user to choose either of them and do a certain task he wants.

The Class contains two attributes of other two important GUI classes which are drawer (The Panel where the user is prompted to draw) and AdvancedColourChoice which is a new frame which is created when the user wants to choose a specific colour not available in the given buttons.

The method getInstance() is done to apply Singleton Design Pattern as it is responsible for creation of only 1 frame of our Canvas Class.



2. Drawer Class

The Drawer class is used to access the JPanel available in the Canvas GUI. It implements the interfaces MouseListener as well as MouseMotionListener to know exactly where the mouse interacts in the Panel. It has many attributes that are set from the Canvas instance. Those attributes are used to decide which functionality is being made in the panel according to the user preference. Depending upon the change in those attributes, the functionality will be decided and acted upon.

Attributes:

1- String shapeName : Stores the name of the shape depending upon the button pressed in the GUI, for instance: SQUARE, CIRCLE, etc.

2- int selectedShape : Stores the index number of the shape selected from the panel. It has an initial value of -1.

3- Color : Stores the color retrieved from the GUI.

4- boolean Filled : States whether the Shape object will be filled or not.

5- boolean selectionMode : States whether the object that will be processed is to be drawn or edited. If selectionMode is false, then we will be drawing an object. Else, an object will be edited.

6- String mode: Stores the mode that will be acted upon, for instance: MOVE, RESIZE, etc.

7- Shape oldShape : Stores a copy of a shape before being processed. It will be used in the undo and redo process.

8- Shape newShape : Stores a copy of a shape after being processed. It will be used in the undo and redo process.

9- ArrayList <Shape> drawings : Stores all shapes that will be drawn on the panel

Methods:

- Accessors and mutators are used for all the above mentioned attributes.

- paintComponent (Graphics g) : The method will loop on all Shapes in the drawings ArrayList and checks the visibility for each. If found true, the Shape would invoke the overridden draw method in his class. The Graphics object 'g' will be set with the color of each Shape object correspondingly.

- mousePressed (MouseEvent me) : This overridden method from MouseListener interface will get the point pressed on the canvas. Depending upon the selectionMode, its behaviour will differ.

  If the selectionMode was found to be false and Shape button was clicked on in the canvas ex: Square, then that means a Shape will be drawn. Depending upon shapeName, the corresponding factory class would be instantiated to create a new Shape. The initial point and the final point for the shape will be set with the point pressed temporarily until a drag occurs. In case of a Triangle instance, points A, B and C will be set with the pressed point. Those points are attributes in the Shapes classes that will be used to draw the objects. The object is then added to the drawings ArrayList. The color and the fill are then set for the object. repaint() is then used to invoke paintComponent method.

  If the selectionMode was found to be true and the drawings ArrayList was found not empty, then that means that the press may result in an edit depending upon the mode. A loop will be made to access the shapes of

drawings and checks using the contains(MyPoint p) method whether the point pressed falls within the boundaries of an existing object. The loop is done from the end so the check can be more accurate in case a bigger shape is drawn on a smaller one. contains method is overridden from the abstract class Shape in all shape classes accordingly to check on the point r. If a shape was found to contain the point, a check for the mode will be done.

If the mode was "MOVE": oldShape would store a copy of the shape before being moved. selectedShape will store the index of that shape. pressedPoint attribute of the Shape will be set with the point pressed. The pressedPoint will be used in the move method of each shape. mouseDragged method is needed thereafter to complete the translation of the shape to the required position.

If the mode was "DELETE": oldShape would store a copy of the shape before being processed. The shape visible attribute will be set with false, and the newShape will then store a copy of the shape after the visibility is set with false. Shapes in drawings are to be set with a false visibility when delete is being made. That is done for the sake of retrieval in case of undo and redo operations.

If the mode was "RESIZE": oldShape would store a copy of the shape before being resized. The overridden resizeShape(MyPoint p) is then invoked. mouseDragged method is needed thereafter to complete the resize of the shape.

If the mode was "CHANGECOLOR": oldShape would store a copy of the shape before being processed. The shape color attribute will be set with color, and the newShape will then store a copy of the shape after the color change.

- mouseDragged (MouseEvent me) : This overridden method from MouseListener interface will get the point of the mouse on the canvas as it is getting dragged. Depending upon the selectionMode, its behaviour will differ.

  If the selectionMode was found to be false and Shape button was clicked on in the canvas ex: Square, then that means a Shape will be drawn. The mouseDragged method is used to determine the finalPoint of the shape. The last element of drawings will be accessed to modify its finalPoint (pointB in case of Triangle). repaint() is then used to invoke paintComponent method.

  If the selectionMode was found to be true and the drawings ArrayList was found not empty, a check will be made on the mode to check whether it is "MOVE" or "RESIZE" and to make sure that selectedShape is not equal to -1.

  In case of "MOVE", the specific shape object will be accessed from drawings using its index (selectedShape) to set its draggedPoint with the point retrieved from the MouseEvent me. The shape would then invoke the moveShape() method of its class. Thereafter, newShape would be set with a copy of the moved shape. repaint() is then used to invoke paintComponent method.

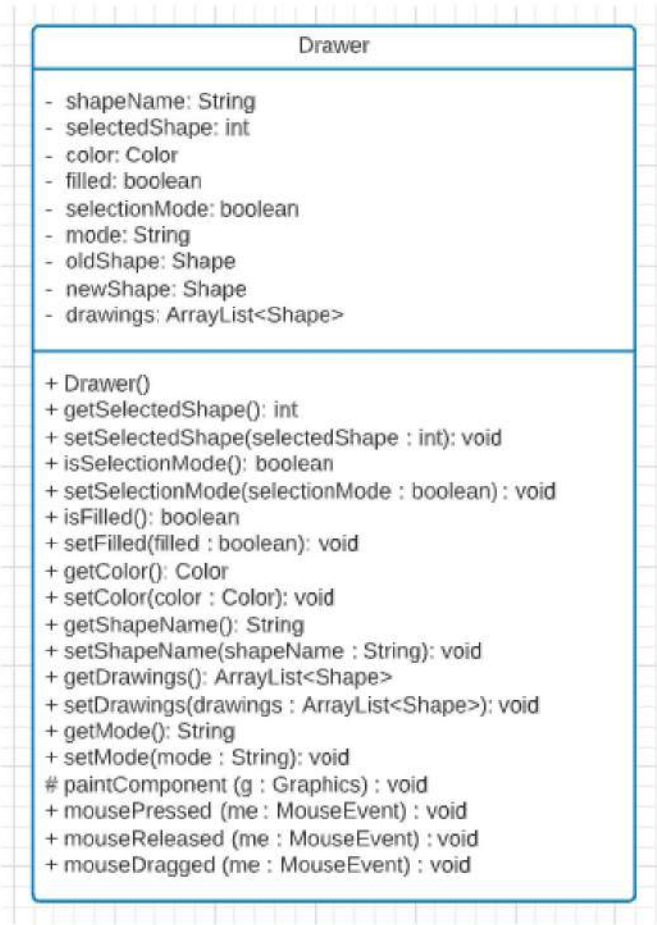
  In case of "RESIZE", the specific shape object will be accessed from drawings using its index (selectedShape) to invoke the resizeShape(My Point p) method of its class with the point from the drag MouseEvent. Thereafter, newShape would be set with a copy of the resized shape.

- mouseReleased(MouseEvent me) : This overridden method from MouseListener interface will act upon the mouse release action either after a press or a drag. It will initialize a new object "l" of the LastAction Class that is used to keep a track of all actions performed on the canvas. This class will

be used for the undo and redo operations. It will check for the mode and accordingly, its behaviour will differ.

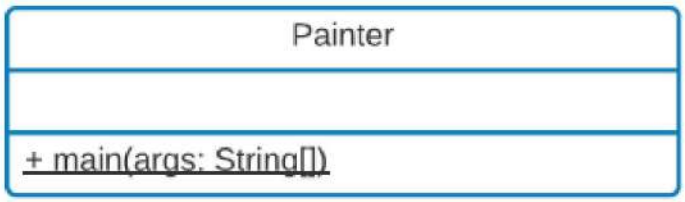
In Case of “DRAW” and “COPY”, oldShape will be set with a copy of the last shape in drawings but with a false visibility. newShape will be set with a copy of that last shape.

In those 2 cases and the others, the LastAction object ‘l’ will be set with the oldShape and newShape and then pushed into the undo stack of the class.



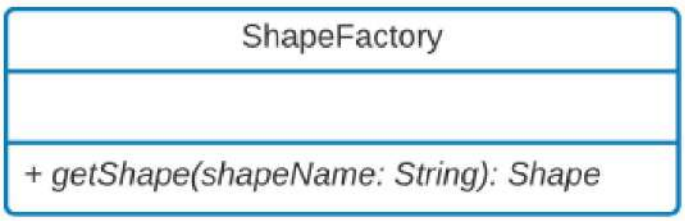
3. Painter Class

This class acts as the driver class for our programme. It initializes an object from the Canvas Frame and sets it to be visible.



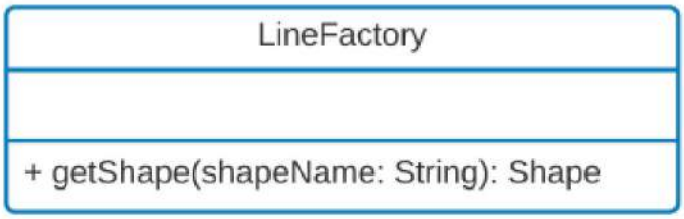
4. ShapeFactory Interface

This interface acts as an Abstract Factory for other several factories. Those factories are LineFactory, EllipticalShapesFactory and PolygonFactory. It contains the abstract method getShape that will be overridden in the in the 3 factories mentioned above.



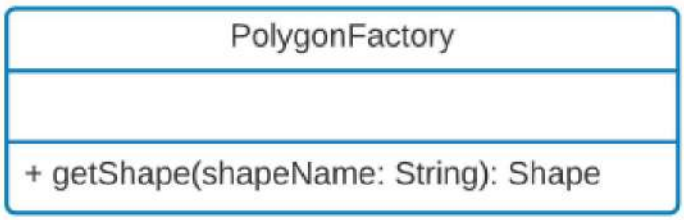
5. LineFactory Class

This class acts as a factory for Line Shape. It implements the Abstract Factory ShapeFactory and thus it overrides the method getShape. When the method is provoked, it checks whether shapeName received is equal to "LINE". If yes, it will initialise a new object of type Line and return it.



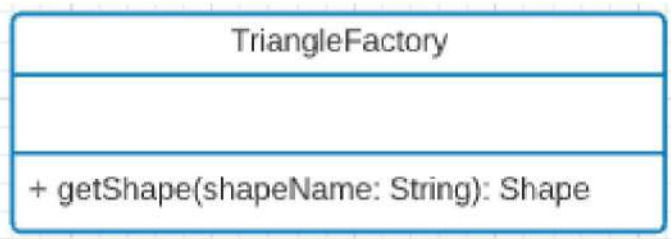
6. PolygonFactory Class

This class acts as a factory for Polygons. The polygons used in our painter application are Squares, Rectangles and Triangles. It implements the Abstract Factory ShapeFactory and thus it overrides the method getShape. When the method is provoked, it checks whether shapeName received is equal to "SQUARE". If yes, it will initialise a new object of type Square and return it. If not, it checks whether it is equal to "RECTANGLE". If yes. , it will initialise a new object of type Rectangle and return it. If the shapeName is either equal to "ISOSCELESTRIANGLE" or "RIGHTANGLETRIANGLE", the method would initialize an object of the TriangleFactory and would invoke its getShape method with the shapeName and return what the TriangleFactory getShape method returns.



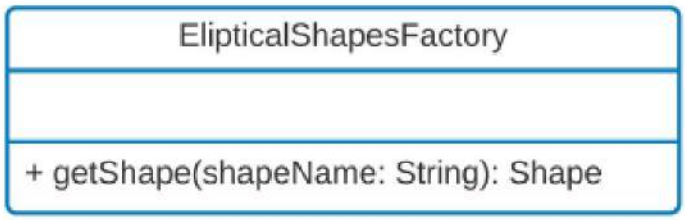
7. TriangleFactory Class

This class extends the PolygonFactory class and overrides its method getShape. It is invoked through the PolygonFactory class and returns either a new object of IsocelesTriangle if the shapeName was equal to "ISOSCELESTRIANGLE" or a new object of RightAngleTriangle if the shapeName was equal to "RIGHTANGLETRIANGLE".



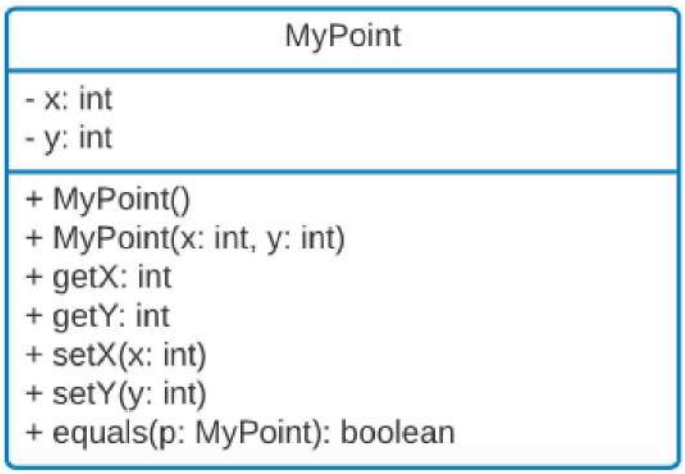
8. EllipticalShapesFactory Class

This class acts as a factory for Elliptical Shapes. It implements the Abstract Factory ShapeFactory and thus it overrides the method getShape. When the method is provoked, it checks whether shapeName received is equal to "ELLIPSE". If yes, it will initialise a new object of type Ellipse and return it. If not, it checks whether shapeName received is equal to "CIRCLE". If yes, it will initialise a new object of type Circle and return it.



9. MyPoint Class

This class is used to mimic a point in the sense that it stores the X-coordinate and the Y-coordinate of it. It has an empty constructor and another that sets the x and y integer attributes. It has accessors and mutators for the x and y. The class has a method equals that receives a MyPoint object and compares is with the x and y attributes and return true if found equal, else it will return false.



10. Shape Class

Shape is an abstract class that is used as a superclass for all the shapes available in our program. It implements Cloneable interface as a clone for Shapes will be needed in case of copying and duplicating.

Attributes:

1- String shapeName : Stores the name of the shape. For instance: SQUARE, CIRCLE, etc.

2- Color : Stores the color to be drawn with of that Shape.

3- boolean Filled : States whether the Shape object will be filled or not.

4- boolean selected : States whether the Shape is was clicked on so that edits will happen to it.

5- boolean visible: States whether the Shape will be displayed on the JPanel or not. It is initialised as true and will be set with false only in case of delete.

6- MyPoint pressedPoint: Stores the point pressed on when wanting to move the Shape.

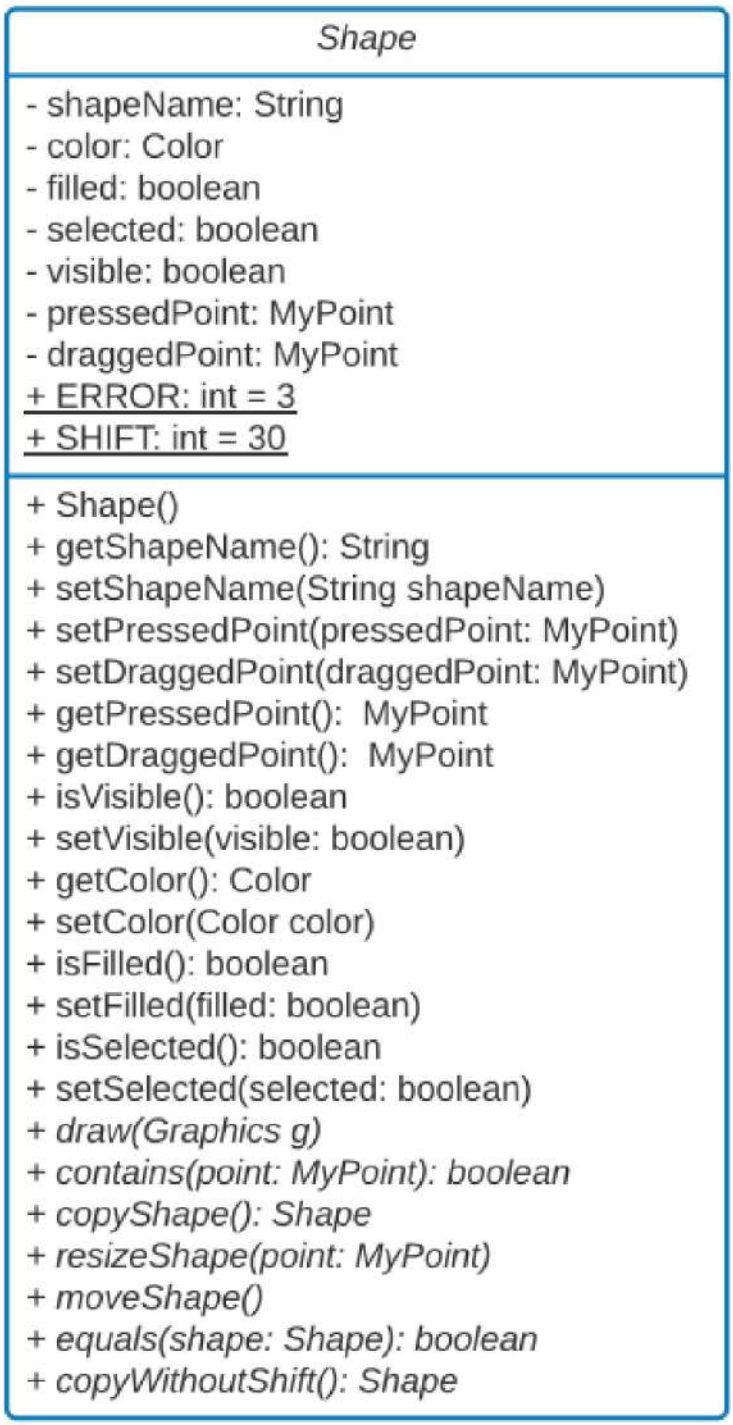
6- MyPoint draggedPoint: Stores the point dragged to when the Shape got moved.

7- int ERROR: Final constant that will be used in the contains method.

8- int SHIFT: Final constant that will be used when duplicating a Shape.

Methods:

- Accessors and mutators are used for all the above mentioned attributes.
- draw(Graphics g) : Is an abstract method that will be overridden in all subclasses. It receives an object of type Graphics that will be used to draw with on the panel.
- contains(MyPoint point) : Is an abstract method that will be overridden in all subclasses. It receives a point in the parameter and checks whether this point falls within the boundaries of the Shape or not. In case of unfilled shapes, it checks whether the click is on the borders or not. It will return true if yes. Else, it will return false.
- copyShape() : Is an abstract method that will be overridden in all subclasses. It returns a shape that is translated with the SHIFT in both X and Y directions.
- copyWithoutShift() : Is an abstract method that will be overridden in all subclasses. It returns an exact clone of the Shape object with the same position.
- moveShape() : Is an abstract method that will be overridden in all subclasses. It is used to translate the Shape into its new position depending upon the pressedPoint and draggedPoint.
- resizeShape(MyPoint point) : Is an abstract method that will be overridden in all subclasses. It sets the final point of the Shape with the point received in the parameter.
- equals(Shape shape) : Is an abstract method that will be overridden in all subclasses. It receives a shape in the parameter and checks whether this shape has the same attributes as the invoker object. If yes, it will return true. Else, it will return false.


Shape
- shapeName: String
- color: Color
- filled: boolean
- selected: boolean
- visible: boolean
- pressedPoint: MyPoint
- draggedPoint: MyPoint
+ ERROR: int = 3
+ SHIFT: int = 30
+ Shape()
+ getShapeName(): String
+ setShapeName(String shapeName)
+ setPressedPoint(pressedPoint: MyPoint)
+ setDraggedPoint(draggedPoint: MyPoint)
+ getPressedPoint(): MyPoint
+ getDraggedPoint(): MyPoint
+ isVisible(): boolean
+ setVisible(visible: boolean)
+ getColor(): Color
+ setColor(Color color)
+ isFilled(): boolean
+ setFilled(filled: boolean)
+ isSelected(): boolean
+ setSelected(selected: boolean)
+ draw(Graphics g)
+ contains(point: MyPoint): boolean
+ copyShape(): Shape
+ resizeShape(point: MyPoint)
+ moveShape()
+ equals(shape: Shape): boolean
+ copyWithoutShift(): Shape

11. Line Class

Attributes:

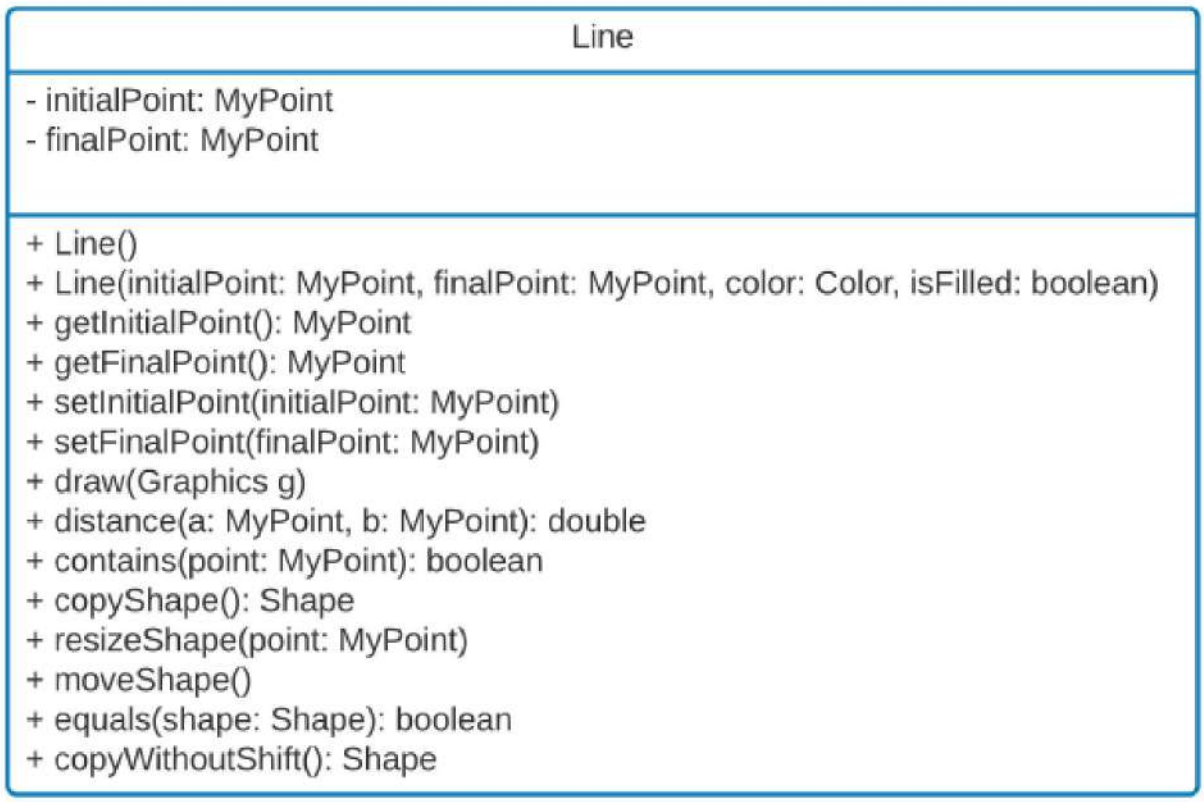
1- MyPoint initialPoint: Stores the starting point of the Line.

2- MyPoint finalPoint: Stores the ending point of the Line.

Methods:

- Accessors and mutators are used for all the above mentioned attributes.
- draw(Graphics g) : This method draws the line on the panel using the method drawLine of the Graphics class. drawLine method will draw the Line using the initialPoint and the finalPoint attributes.
- distance(MyPoint a, MyPoint b) : this method returns the distance between points a and b using the Pythagorean Theorem.
- contains(MyPoint point) : this method checks whether the point is on the line or not. It makes the check by calculating the distance between the initialPoint and point and adding it to the distance between the point and the finalPoint. If found equal to the distance between the initialPoint and the finalPoint with a small difference (ERROR) then it will return true.
- copyShape() : this method will use the clone method of the Cloneable interface to make an exact copy of the Line. The cloned line will then be translated with the SHIFT in both X and Y directions. The new deep-copied line will then be returned.
- copyWithoutShift() : this method will perform the same actions as copyShape method but will return an unshifted Line.
- resizeShape(MyPoint point): this method will resize the shape by changing the finalPoint to point.
- moveShape() : this method will get the difference between the pressedPoint and draggedPoint and translate the Line to the new position accordingly.

- equals() : checks that the received Line object equals the Line invoker object in all the attributes.



12. Square Class

Attributes:

1- MyPoint initialPoint: Stores the starting point of the Square.

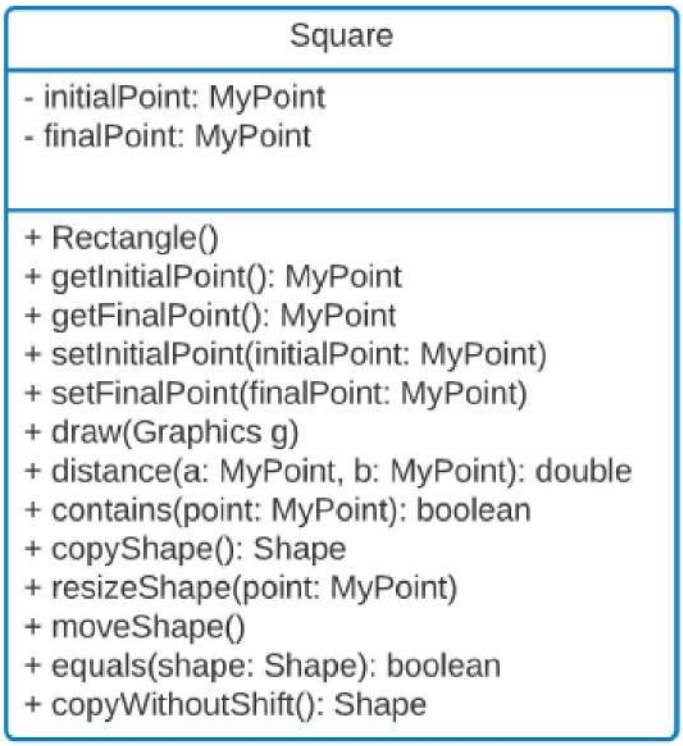
2- MyPoint finalPoint: Stores the ending point of the Square.

Methods:

- Accessors and mutators are used for all the above mentioned attributes.
- draw(Graphics g) : This method draws the Square on the panel using the method drawRect or fillRect of the Graphics class. The method will draw the Square using the initialPoint and the width and height which are equal in our case. According to which quadrant will be drawing in, the method will be invoked differently. If the filled attribute was found to be true, the fillRect is invoked. Else, the drawRect is invoked.
- distance(int x1, int y1, int x2, int y2) : this method returns the distance between points 1 and 2 using the Pythagorean Theorem.
- contains(MyPoint point) : this method checks whether the point lies within the boundaries of the Square in case it was filled or on the borders of the Square in case of unfilled. To check whether the point is inside the Square or not, we check that the X-Coordinate of the point is within the range of the X-Coordinate of the initialPoint and the X-Coordinate of the initialPoint + side length and same for the Y-Coordinate. Implementation will differ according to which quadrant the object was drawn in. To check whether the point is on the borders or not, we use the same method as in line by checking that the distance between the vertex and point plus distance between point and another vertex is the same as distance between the 2 vertices.
- copyShape() : this method will use the clone method of the Cloneable interface to make an exact copy of the Square. The cloned Square will then

be translated with the SHIFT in both X and Y directions. The new deep-copied Square will then be returned.

- copyWithoutShift() : this method will perform the same actions as copyShape method but will return an unshifted Square.
- resizeShape(MyPoint point): this method will resize the shape by changing the finalPoint to point.
- moveShape() : this method will get the difference between the pressedPoint and draggedPoint and translate the Square to the new position accordingly.
- equals() : checks that the received Square object equals the Square invoker object in all the attributes.



13. Rectangle Class

Attributes:

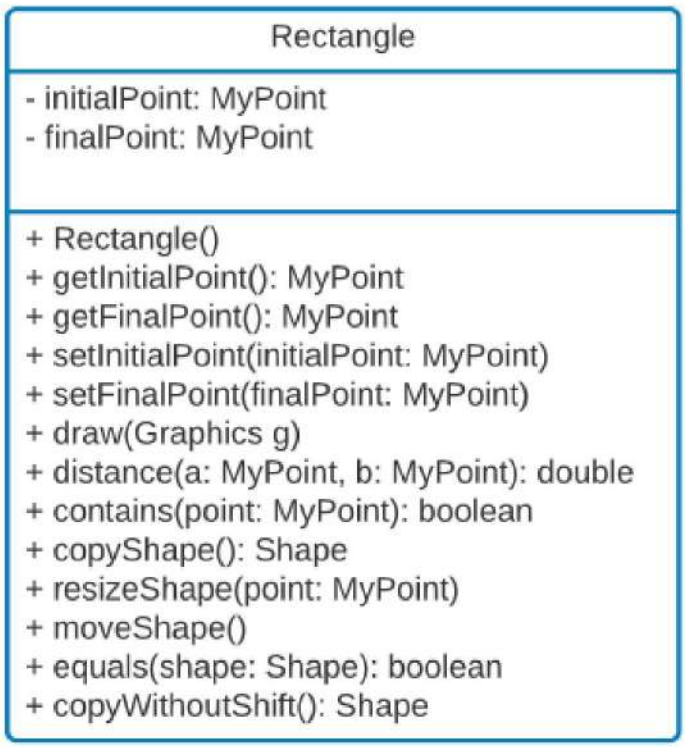
1- MyPoint initialPoint: Stores the starting point of the Rectangle.

2- MyPoint finalPoint: Stores the ending point of the Rectangle.

Methods:

- Accessors and mutators are used for all the above mentioned attributes.
- draw(Graphics g) : This method draws the Rectangle on the panel using the method drawRect or fillRect of the Graphics class. The method will draw the Rectangle using the initialPoint and the width and height which will be calculated from the difference between the initialPoint and the finalPoint. According to which quadrant will be drawing in, the method will be invoked differently. If the filled attribute was found to be true, the fillRect is invoked. Else, the drawRect is invoked.
- distance(int x1, int y1, int x2, int y2) : this method returns the distance between points 1 and 2 using the Pythagorean Theorem.
- contains(MyPoint point) : this method checks whether the point lies within the boundaries of the Rectangle in case it was filled or on the borders of the Rectangle in case of unfilled. To check whether the point is inside the Rectangle or not, we check that the X-Coordinate of the point is within the range of the X-Coordinate of the initialPoint and the X-Coordinate of the initialPoint + width and the Y-Coordinate is within the range of the Y-Coordinate of the initialPoint and the Y-Coordinate of the initialPoint + height. Implementation will differ according to which quadrant the object was drawn in. To check whether the point is on the borders or not, we use the same method as in line by checking that the distance between the vertex and point plus distance between point and another vertex is the same as distance between the 2 vertices.

- copyShape() : this method will use the clone method of the Cloneable interface to make an exact copy of the Rectangle. The cloned Rectangle will then be translated with the SHIFT in both X and Y directions. The new deep-copied Rectangle will then be returned.
- copyWithoutShift() : this method will perform the same actions as copyShape method but will return an unshifted Rectangle.
- resizeShape(MyPoint point): this method will resize the shape by changing the finalPoint to point.
- moveShape() : this method will get the difference between the pressedPoint and draggedPoint and translate the Rectangle to the new position accordingly.
- equals() : checks that the received Rectangle object equals the Rectangle invoker object in all the attributes.



14. Triangle Class

This class acts as the superclass for the classes RightAngleTriangle and IsoscelesTriangle.

Attributes:

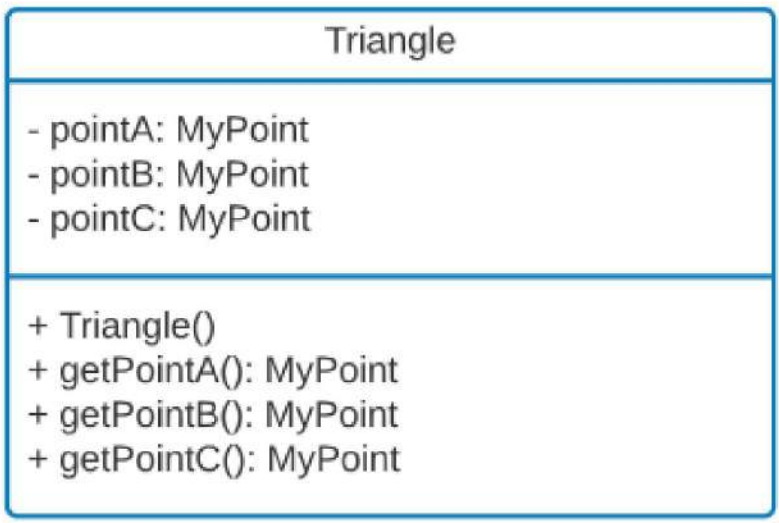
1- MyPoint pointA: Stores the first vertex of the Triangle.

2- MyPoint pointB: Stores the second vertex of the Triangle.

3- MyPoint pointC: Stores the third vertex of the Triangle.

Methods:

- Accessors and mutators are used for all the above mentioned attributes.

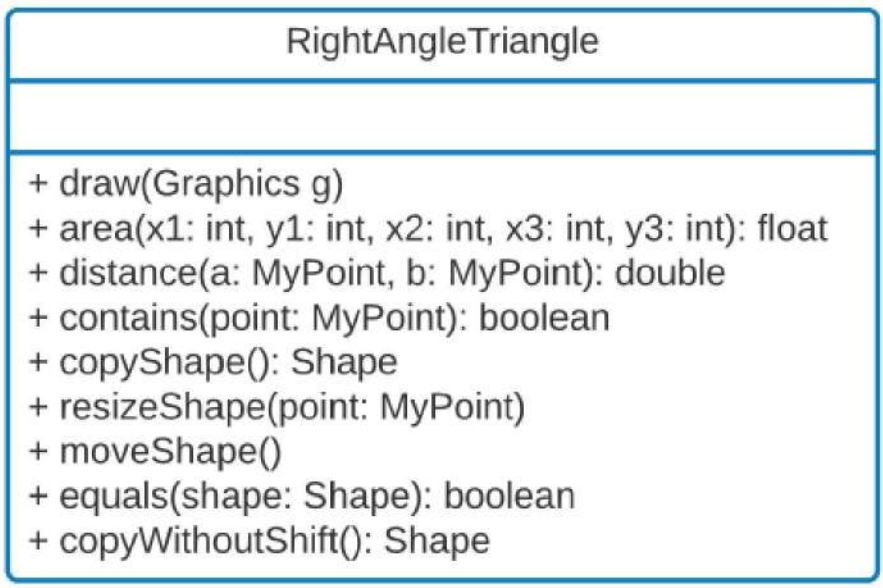


15. RightAngleTriangle Class

Methods:

- draw(Graphics g) : This method will draw the Right angled triangle on the panel using the method drawPolygon or fillPolygon of the Graphics class. The method will draw the triangle using the pointA, pointB and pointC. If the filled attribute was found to be true, then the fillPolygon is invoked. Else, the drawPolygon is invoked.
- area(int x1, int y1, int x2, int y2, int x3, int y3) : this method is used to calculate the area of the triangle and returns it as float.
- distance(MyPoint a, MyPoint b) : this method returns the distance between points a and b using the Pythagorean Theorem.
- contains(MyPoint point) : this method checks whether the point lies within the boundaries of the Triangle in case it was filled or on the borders in case of unfilled. To check whether the point is inside the Triangle or not, we calculate the area of a triangle joining the point with pointA and pointB and add it to the area of a triangle joining the point to pointB and pointC and add it to the area of the triangle joining the point to pointC and pointA. If the sum of those 3 areas equal the total area of the triangle, then that point is within the filled triangle. To check whether the point is on the borders or not, we use the same method as in line by checking that the distance between the vertex and point plus distance between point and another vertex is the same as distance between the 2 vertices.
- copyShape() : this method will use the clone method of the Cloneable interface to make an exact copy of the RightAngleTriangle. The cloned RightAngleTriangle will then be translated with the SHIFT in both X and Y directions. The new deep-copied RightAngleTriangle will then be returned.
- copyWithoutShift() : this method will perform the same actions as copyShape method but will return an unshifted RightAngleTriangle.

- resizeShape(MyPoint point): this method will resize the shape by changing pointB to point.
- moveShape() : this method will get the difference between the pressedPoint and draggedPoint and translate the RightAngleTriangle to the new position accordingly.
- equals() : checks that the received RightAngleTriangle object equals the RightAngleTriangle invoker object in all the attributes.

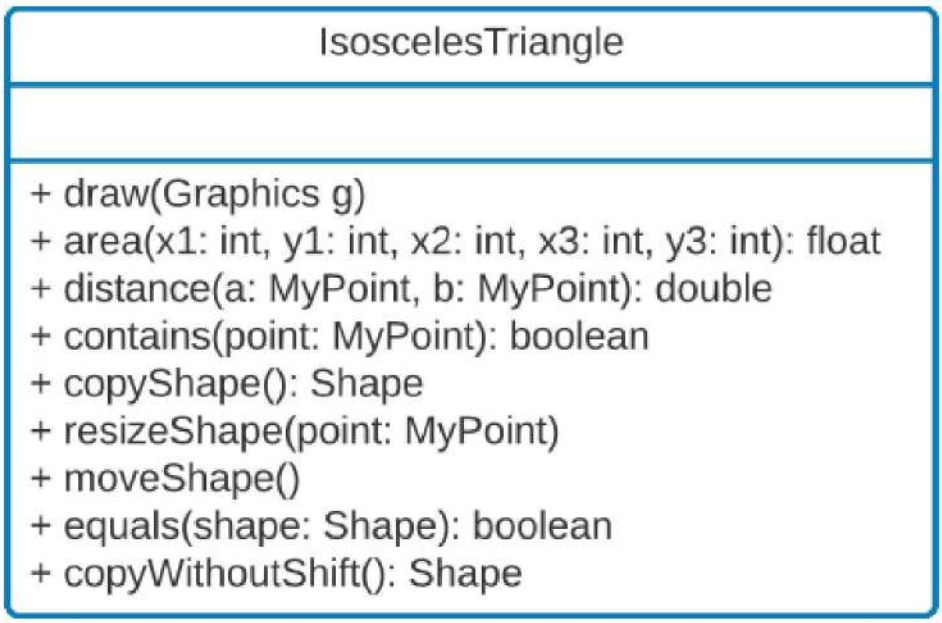


16. IsoscelesTriangle Class

Methods:

- draw(Graphics g) : This method will draw the Isosceles triangle on the panel using the method drawPolygon or fillPolygon of the Graphics class. The method will draw the triangle using the pointA, pointB and pointC. If the filled attribute was found to be true, then the fillPolygon is invoked. Else, the drawPolygon is invoked.
- area(int x1, int y1, int x2, int y2, int x3, int y3) : this method is used to calculate the area of the triangle and returns it as float.
- distance(MyPoint a, MyPoint b) : this method returns the distance between points a and b using the Pythagorean Theorem.
- contains(MyPoint point) : this method checks whether the point lies within the boundaries of the Triangle in case it was filled or on the borders in case of unfilled. To check whether the point is inside the Triangle or not, we calculate the area of a triangle joining the point with pointA and pointB and add it to the area of a triangle joining the point to pointB and pointC and add it to the area of the triangle joining the point to pointC and pointA. If the sum of those 3 areas equal the total area of the triangle, then that point is within the filled triangle. To check whether the point is on the borders or not, we use the same method as in line by checking that the distance between the vertex and point plus distance between point and another vertex is the same as distance between the 2 vertices.
- copyShape() : this method will use the clone method of the Cloneable interface to make an exact copy of the IsoscelesTriangle. The cloned IsoscelesTriangle will then be translated with the SHIFT in both X and Y directions. The new deep-copied IsoscelesTriangle will then be returned.
- copyWithoutShift() : this method will perform the same actions as copyShape method but will return an unshifted IsoscelesTriangle.

- resizeShape(MyPoint point): this method will resize the shape by changing pointB to point.
- moveShape() : this method will get the difference between the pressedPoint and draggedPoint and translate the IsoscelesTriangle to the new position accordingly.
- equals() : checks that the received IsoscelesTriangle object equals the IsoscelesTriangle invoker object in all the attributes.



17. Ellipse Class

Attributes:

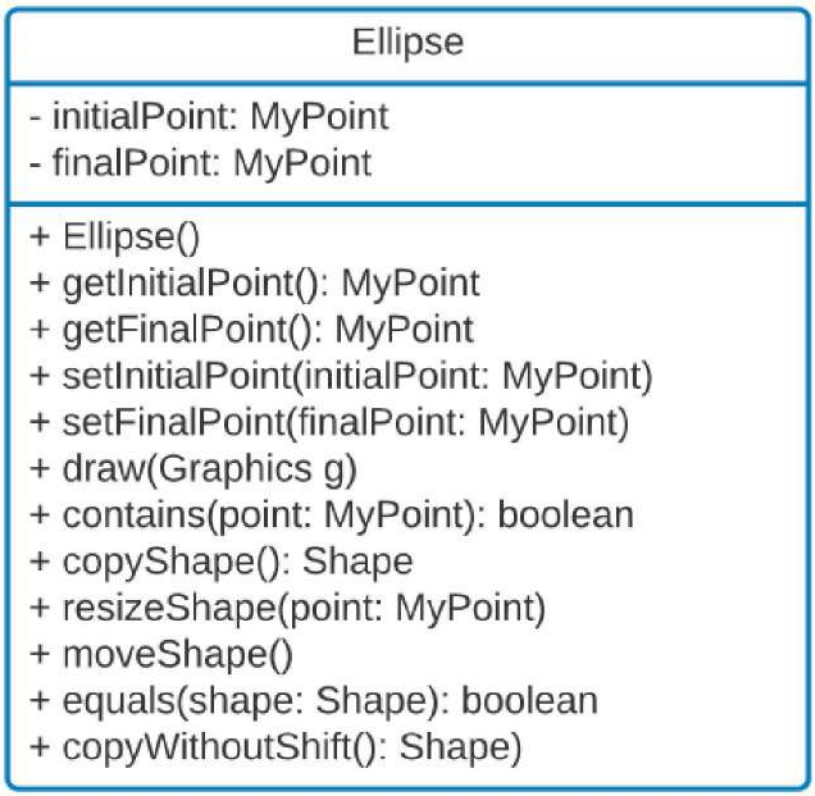
1- MyPoint initialPoint: Stores the starting point of the Ellipse.

2- MyPoint finalPoint: Stores the ending point of the Ellipse.

Methods:

- Accessors and mutators are used for all the above mentioned attributes.
- draw(Graphics g) : This method draws the Ellipse on the panel using the method drawOval or fillOval of the Graphics class. The method will draw the Ellipse using the initialPoint and the width and height which will be calculated from the difference between the initialPoint and the finalPoint. According to which quadrant will be drawing in, the method will be invoked differently. If the filled attribute was found to be true, the fillOval is invoked. Else, the drawOval is invoked.
- contains(MyPoint point) : this method checks whether the point lies within the boundaries of the Ellipse in case it was filled or on the borders of the Ellipse in case of unfilled. To check whether the point is inside the Ellipse or not, the ellipse equation will be used and accordingly the point will be compared to it. The point must satisfy the equation in case if It was not filled. Implementation will differ according to which quadrant the object was drawn in.
- copyShape() : this method will use the clone method of the Cloneable interface to make an exact copy of the Ellipse. The cloned Ellipse will then be translated with the SHIFT in both X and Y directions. The new deep-copied Ellipse will then be returned.
- copyWithoutShift() : this method will perform the same actions as copyShape method but will return an unshifted Ellipse.

- resizeShape(MyPoint point): this method will resize the shape by changing the finalPoint to point.
- moveShape() : this method will get the difference between the pressedPoint and draggedPoint and translate the Ellipse to the new position accordingly.
- equals() : checks that the received Ellipse object equals the Ellipse invoker object in all the attributes.



18. Circle Class

Attributes:

1- MyPoint initialPoint: Stores the starting point of the Circle.

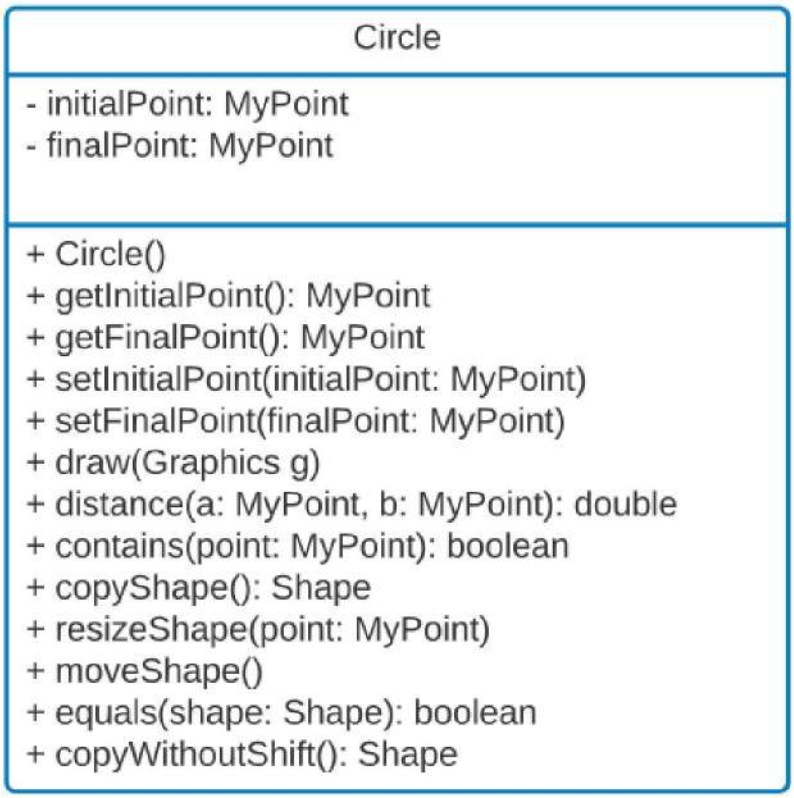
2- MyPoint finalPoint: Stores the ending point of the Circle.

Methods:

- Accessors and mutators are used for all the above mentioned attributes.
- draw(Graphics g) : This method draws the Circle on the panel using the method drawOval or fillOval of the Graphics class. The method will draw the Circle using the initialPoint and the width and height which will be equal in our case. According to which quadrant will be drawing in, the method will be invoked differently. If the filled attribute was found to be true, the fillOval is invoked. Else, the drawOval is invoked.
- distance(MyPoint a, MyPoint b) : this method returns the distance between points a and b using the Pythagorean Theorem.
- contains(MyPoint point) : this method checks whether the point lies within the boundaries of the Circle in case it was filled or on the borders of the Circle in case of unfilled. To check whether the point is inside the Circle or not, the center of the circle will need to be calculated using the initialpoint. The distance between the center and the point should be less than or equal to the radius, If so then the method will return true. In case that the Circle was unfilled, the distance between the center and the point must be equal to the radius. Implementation will differ according to which quadrant the object was drawn in.
- copyShape() : this method will use the clone method of the Cloneable interface to make an exact copy of the Circle. The cloned Circle will then be

translated with the SHIFT in both X and Y directions. The new deep-copied Circle will then be returned.

- copyWithoutShift() : this method will perform the same actions as copyShape method but will return an unshifted Circle.
- resizeShape(MyPoint point): this method will resize the shape by changing the finalPoint to point.
- moveShape() : this method will get the difference between the pressedPoint and draggedPoint and translate the Circle to the new position accordingly.
- equals() : checks that the received Circle object equals the Circle invoker object in all the attributes.



19. LastAction Class

This class is done solely for the sake of recording all actions performed on the canvas.

Attributes:

1- Shape oldShape: Stores the old state of a Shape.

2- Shape newShape: Stores the new state of a Shape.

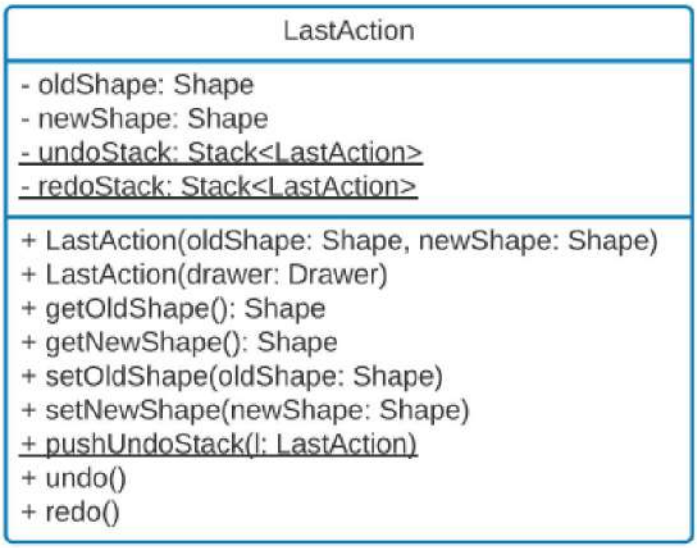
3- Stack<LastAction> undoStack: It is used to store all the actions happening on the canvas with the old state and the new state.

4- Stack<LastAction> redoStack: It is used to store LastAction instances popped from the undoStack after the undo operation was made. It will be used in the redo operation.

Methods:

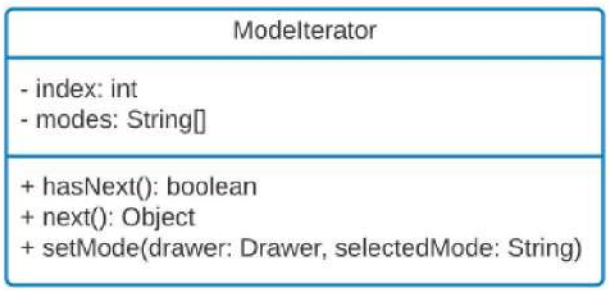
- Accessors and mutators are used for all the above mentioned attributes.
- pushUndoStack(LastAction l) : This method receives an object of the LastAction class and pushes it in the undoStack. This method is invoked after every action is performed.
- undo(): This method checks that the undoStack is not empty and then pops out LastAction object. It will loop on the drawings ArrayList to compare the newShape of the popped object with the Shapes in drawings. Once a match is found, the Shape in the ArrayList will be replaced by the oldShape of the popped object. A new LastAction will be made with popped object newShape as the oldShape and the popped object oldShape as the newShape. This object will then be pushed to the redoStack. The repaint method is then invoked.
- redo(): This method checks that the redoStack is not empty and then pops out LastAction object. It will loop on the drawings ArrayList to compare the newShape of the popped object with the Shapes in drawings. Once a match is found, the Shape in the ArrayList will be replaced by the oldShape of the popped object. A new LastAction will be made with popped object

newShape as the oldShape and the popped object oldShape as the newShape. This object will then be pushed to the undoStack.



20. ModeIterator Class

This class implements the iterator interface and overrides its methods next() and hasNext(). It is used to set the mode of the drawer depending upon the selectedMode parameter in the setMode method. The setMode method will loop on the modes array and compare. Upon finding a match, the array element will be assigned to the drawer mode. The loop uses the next() and hasNext() methods.



Design Patterns

1- Abstract Factory:

This design pattern is responsible for creating objects of the shapes drawn by the user by the help of other concrete factories.

In our case we have ShapeFactory interface that implements the Abstract Factory design pattern and the concrete factories are PolygonFactory,EllipticalShapes Factory, TriangleFactory, LineFactory.

2- Factory:

This Design pattern is responsible for creating each object according to the input taken from the Abstract Factory then is sent to the concrete classes which directly creates the desired shape.

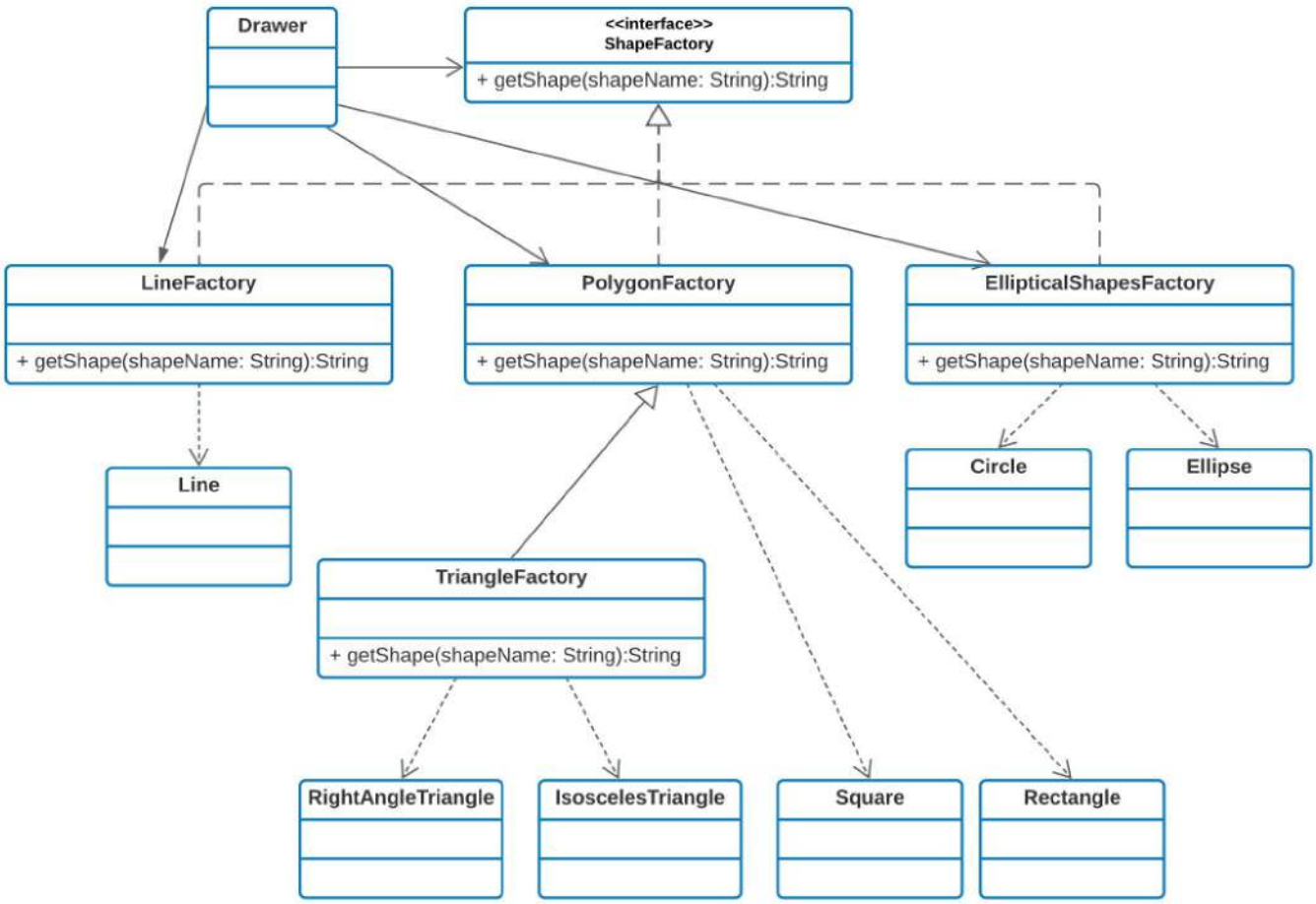
PolygonFactory: Construct and returns an object of Square, Rectangle or Triangle using the TriangleFactory.

TriangleFactory: Construct and returns an object of RightAngleTriangle or IsoscolesTriangle.

EllipticalShapes Factory: Construct and returns an object of Circle or Ellipse.

LineFactory: Construct and returns an object of Line.

➔ The following class diagrams illustrates the previous Design patterns.



3- Singleton:

This allows the user to create a single object of a certain class, where the system is designed to have one object only from that class.

This is implemented in our program using the Canvas class which implements one Drawer only.

4- Prototype:

This helps the user to copy an object with the same attributes by cloning it.

the clone method of the cloneable interface constructs a new object of type Shape that will be casted into the desired shape type and returns it.

This is implemented by the methods copyShape(),copyWithoutShift() and paste().

5- Iterator:

This pattern is used to get a way to access the elements of the available modes as DRAW and MOVE in sequential manner without any need to know its underlying representation.

When the user clicks on a button a ModeIterator object is created and the object loops on the array of modes to set the mode by the required one.

6- Composite:

This pattern creates a class that contains group of its own objects. This class provides ways to modify its group of same objects.

That is implemented by the class LastAction that has a stack of objects of type LastAction.

7- Memento:

Memento pattern is used to restore state of an object to a previous state.

Before doing any action on a certain shape a copy of that shape is stored with all the attributes in a stack to help the restoring of the object keeping its previous attributes when undoing the operation done.

8- Mediator:

Mediator pattern is used to reduce communication complexity between multiple objects or classes. This pattern provides a mediator class which normally handles all the communications between different classes and supports easy maintenance of the code by loose coupling.

That is implemented in our ModeIterator class which is considered the mediator class between the Canvas and the Drawer.

OOD Principles

S.O.L.I.D

1- SRP: Every class in our programme is responsible for only one duty.

For example: ModeIterator Class is only responsible for returning the current mode of our programme.

2- OCP: Every class in our programme is open for extension but closed for modification.

For example: Our Abstract class Shape have abstract methods that are unmodifiable which are overriden later on each class that extends Shape, so if you need to add another type of shapes you just extends the class Shape and implement the abstract methods which satisfies the desires of that new shape.

3- LSP: Subclasses in our programme are substitutable for their superclass without any disturbance in the behavior.

For example: RightAngleTriangle Class and IsoscelesTriangle Class are substitutable for the Triangle Class.

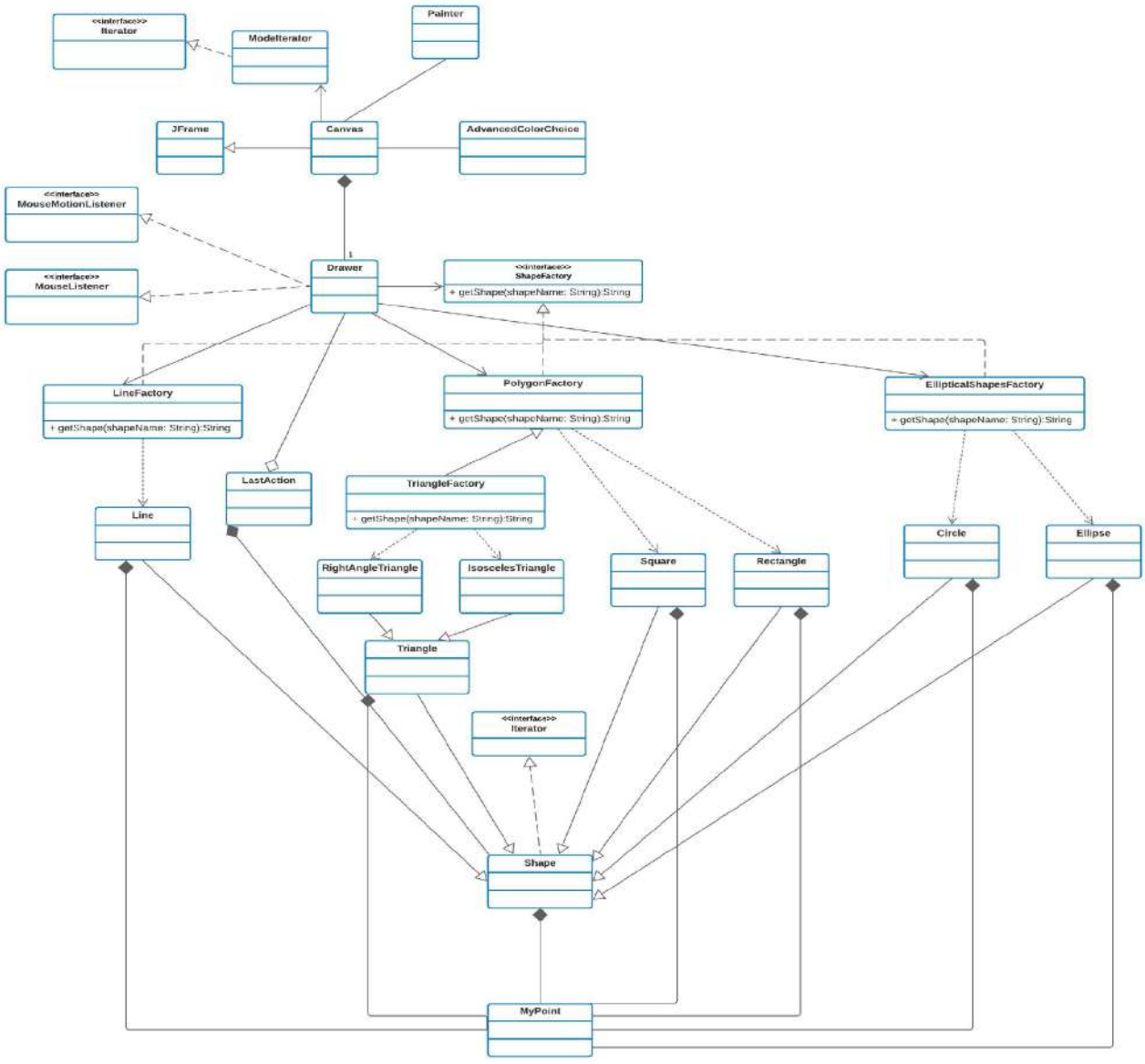
4- ISP: The interface in our programme is very specified to avoid unnecessary dependencies.

For example: our ShapeFactory Interface have only one method that gets a shape by the usage of other factories.

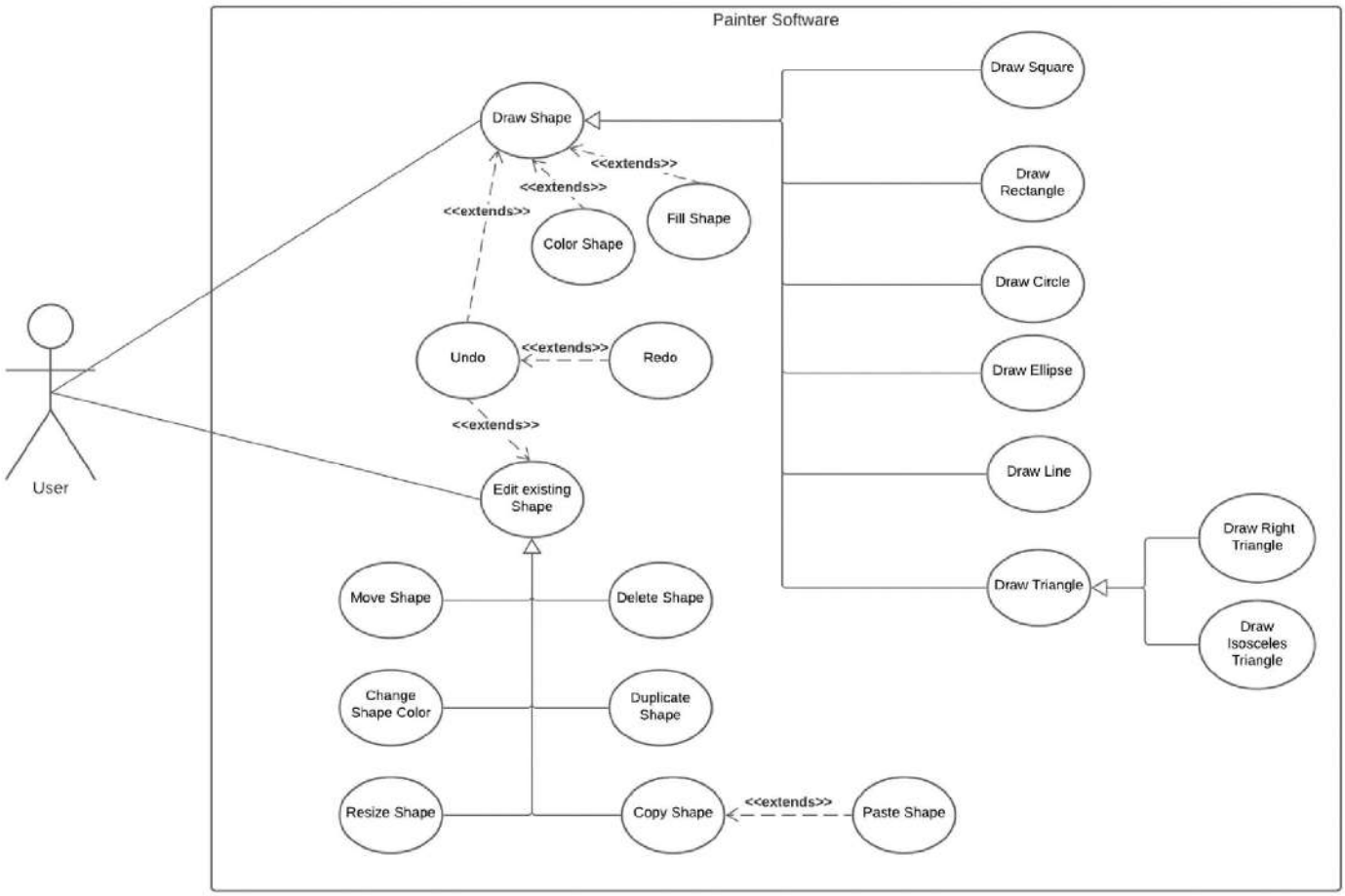
5- DIP: The creational design of our programme implements that principle as the construction of any shape is done using the abstract factory by the help of the other concrete factories.

UML Diagrams

a) Class Diagrams (to show relationships between classes)



b) Use Case Diagram



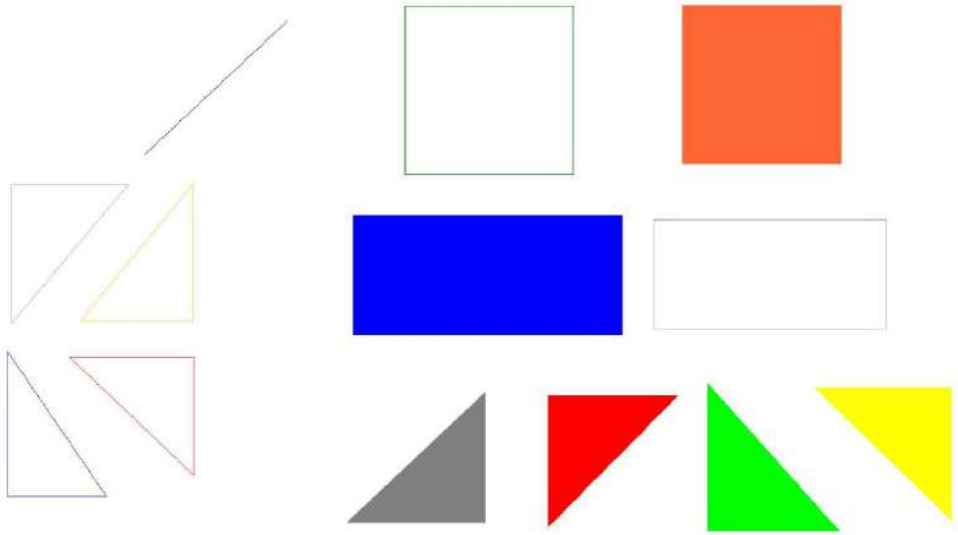
Sample Runs

Video Tutorial:

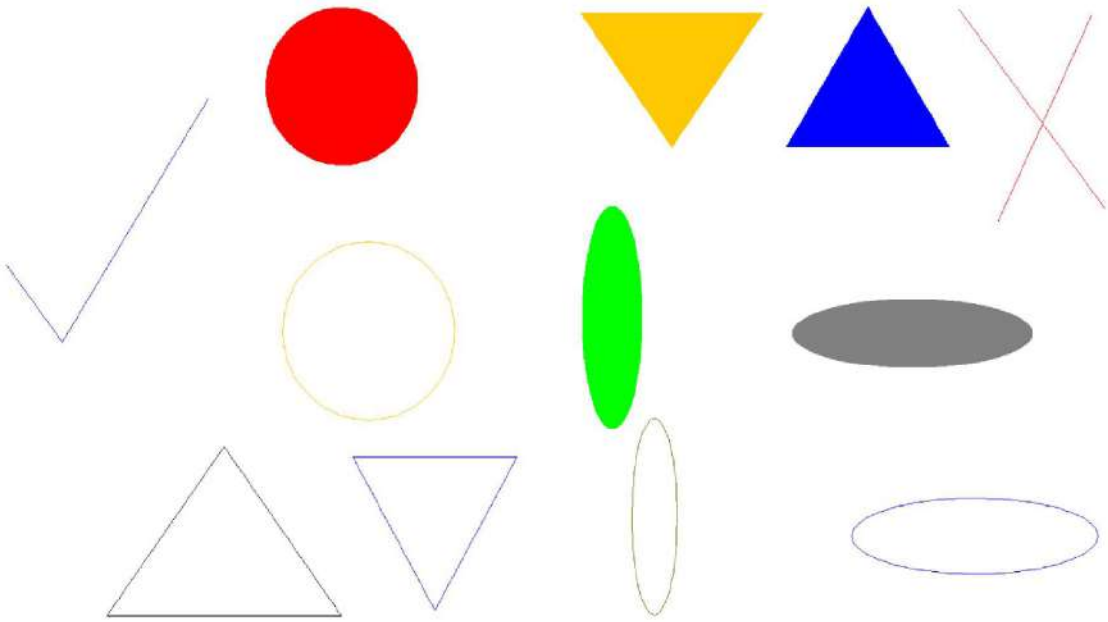
https://drive.google.com/file/d/1XKTbDCvpmtqr4MZqyIXBjnvlU6lJTNL9/view?usp=sharing

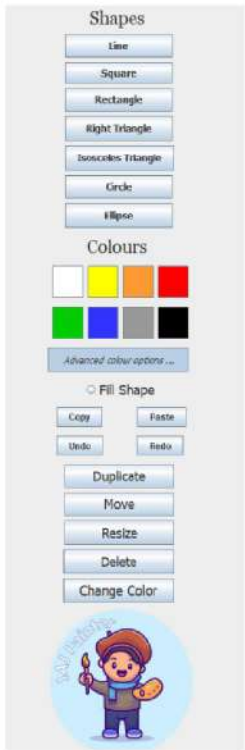
- Here our programme main interface is shown

- You can draw any of the available shapes or edit any existing shape



- A shape could be colored before drawing or after and you either choose to draw a filled or unfilled one.





You can move, delete, resize or duplicate a shape.

- In addition you can copy a shape and paste it in another place of your choice.