Painting Program

Islam Ahmed Bassuni *7229*

Abdallah Gamal Mubarek *6771*

Basel Ashraf Elkarray *6705*

Marwan Ashraf Abdelnaby *7044*

Introduction:

It’s a matter of fact that painting programs are essential for any desktop personal computer. This report discusses our implementation of paint program using java programming language and its concepts of object-oriented design. Our program depends mainly on using JavaFx library which is a java library that is used to make 2D graphics.

Packages:

1. Canvas:
   a. Main class:

      This class extends **Application** class which will be used in running the program. The class prepares the canvas and the stage needed for the painting space. Moreover, it specifies the dimensions of canvas and the title. It also launches the program using **launch** method.

b. Logic class:

This class contains the implementation of the main methods used inside the program. The class contains the following methods:

1. openGettingStarted:

Displayes the user guide for the user

2. setFillYes: sets the value of the member color using ColorPicker library. It is also related to the fxml file

3. Methods for setting the color of filled shape or the stroke of transparent shape using ColorPicker library

4. Methods for setting the mode of the program (copy, move, delete, square, circle, rectangle, straight line, scribble, ellipse, closed polygon)

5. clearCanvas method for clearing the canvas after drawing

6. startDraw method: this method is mainly responsible for creating the shapes and coloring them with the chosen color. It also sets the shape dimensions and coordinates

7. drag method: this method is responsible for tracing mouse movement

8.endDraw: this method stores the drawn shapes in the stack in which it will be used in undo or redo

9. undo method: this method is responsible with dealing with the stack of shapes in which it stores them and restores the state of the canvas while catching the event on undoing.

10.redo is responsible of popping from the stack the restores shapes

11. delete: this method is responsible for handling the deletion action by the user.

c. Canvas fxml file:

this file contains the created GUI by using scene builder using javafx

2. Shapes Models:

This package contains the shapes models in which all the shapes classes inherit from the class MyShape. It also contains the class ShapeFactory in which it is used to implement factory design pattern.

3. Action package:

This package contains the class Clear which is used n clearing the canvas after finishing painting.

Design Patterns:

1. Factory design pattern:

we applied this design pattern using the class **ShapeFactory** in which shapes instances are created. we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

2. Singleton design pattern:

This pattern involves a single class which is responsible to create an object while making sure that

only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class. We applied this pattern through making the only one instance of the class **clear** and we return this instance through getInstance() method.

3. Iterator design pattern:

This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation. We applied this pattern through importing (java.util.Iterator) and using it in the undo method in which we create iterato to iterate through created objects in the undoHistory stack.

4. Composite design pattern:

Composite pattern is a partitioning design pattern and describes a group of objects that is treated the same way as a single instance of the same type of object. We applied this design pattern through creating a stack of instances of the class **MyShape.**

5. Template design pattern:

Template method design pattern is to define an algorithm as a skeleton of operations and leave the details to be implemented by the child classes. We implemented this pattern through creating abstract method (draw()) in the class Shape, then implemented these methods in the subclasses.

6. Bridge design pattern:

We applied this pattern through **clear** class which contains the implementation of clearing the canvas and this class inherits from the MyShape class. So we are separating the implementation from the class MyShape.

Solid Principles:

1. Single-responsibility Principle:

This principle states that a class should have one and only one reason to change, meaning that a class should have only one job. We applied this concept by making modification of class separated from each other.

2. Open close principle:

This concept states that objects or entities should be open for extension but closed for modification. We applied this concept by making inheritance in the MyShape class in which all classes of shapes inherit from this class.

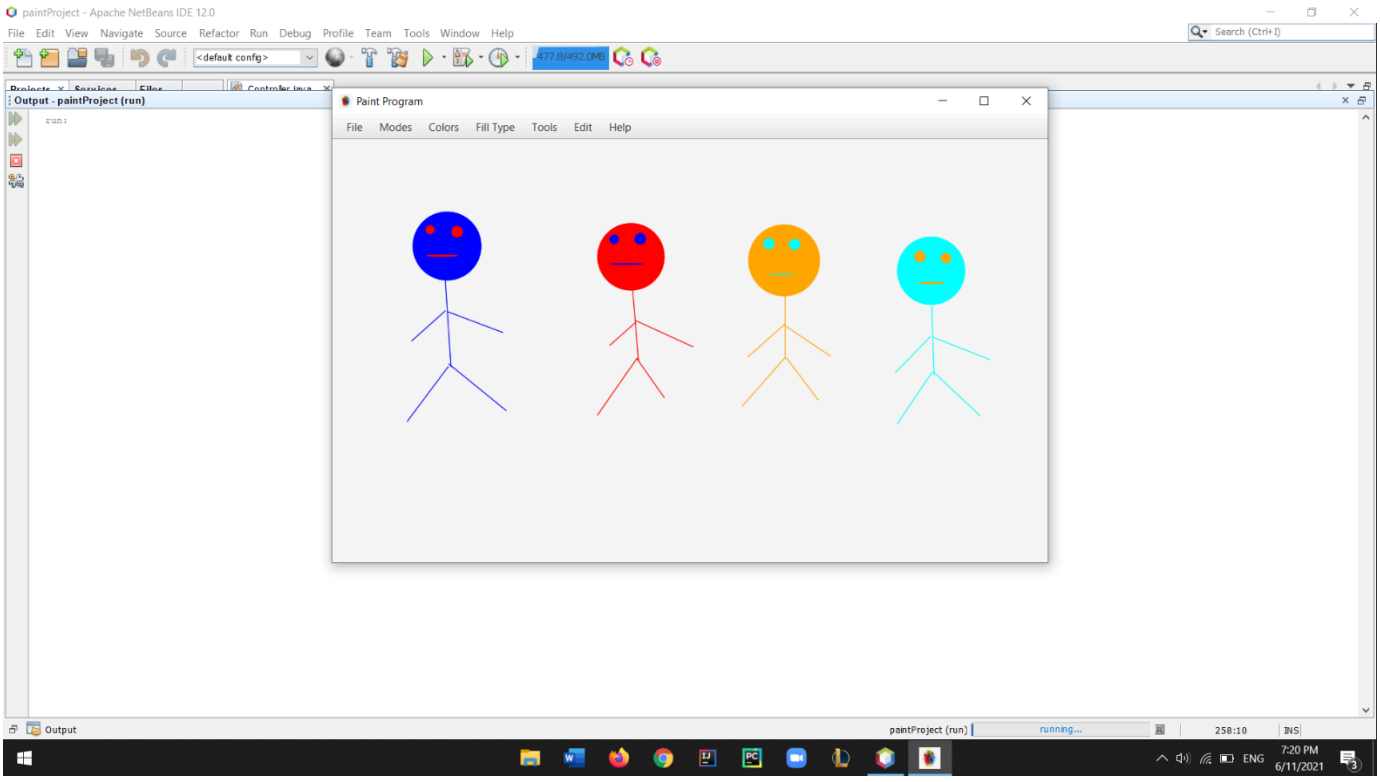
3. Liskov Substitution Principle:

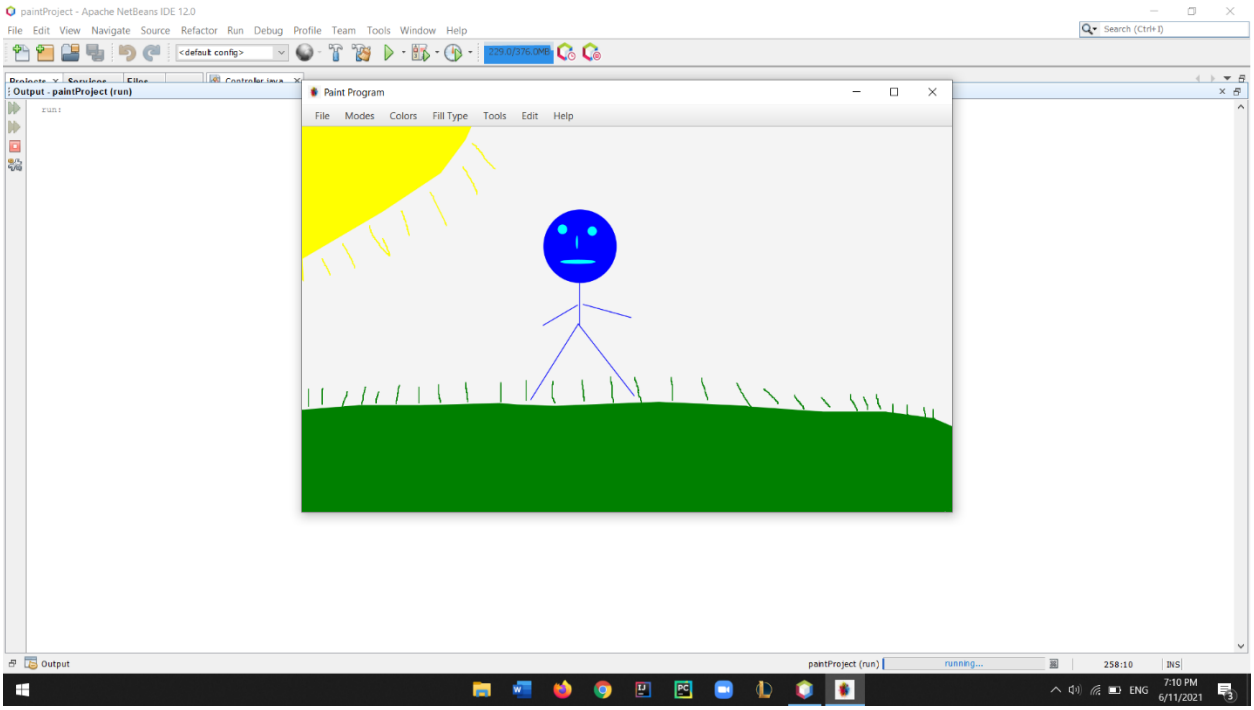
This concept states that every subclass or derived class should be substitutable for their base or parent class. We applied this principle through making subclasses efficient as possible so that subclasses can act with the same functionality of parent class

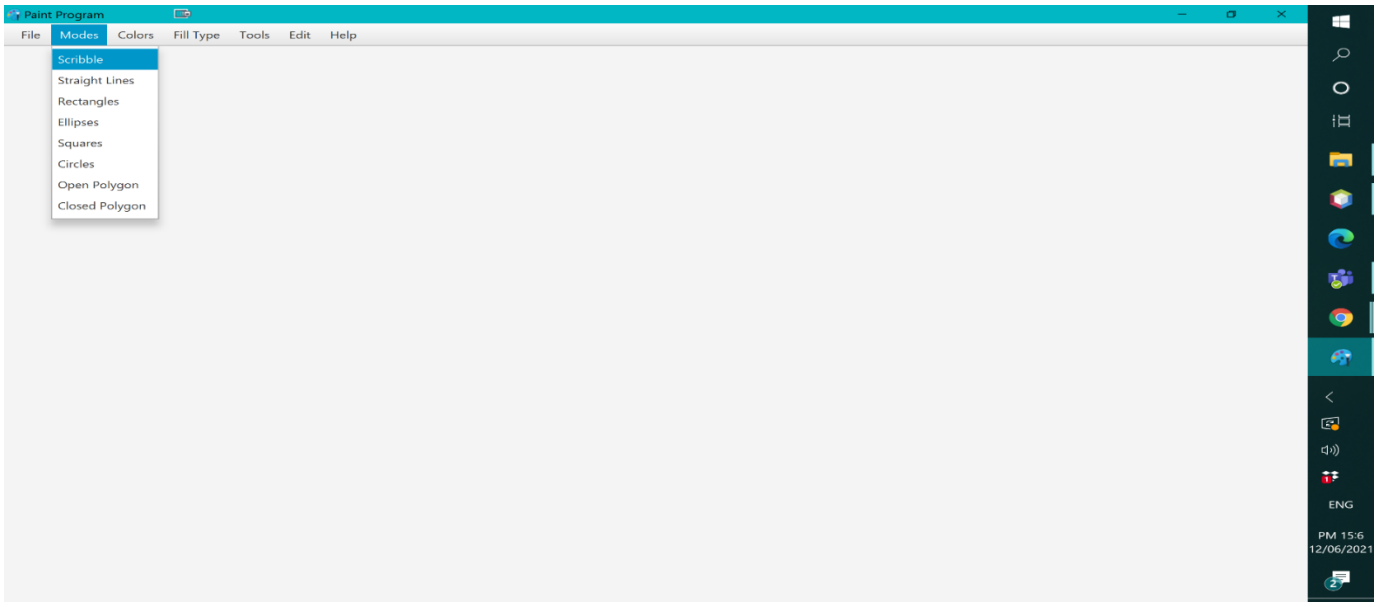
4. Interface Segregation Principle:

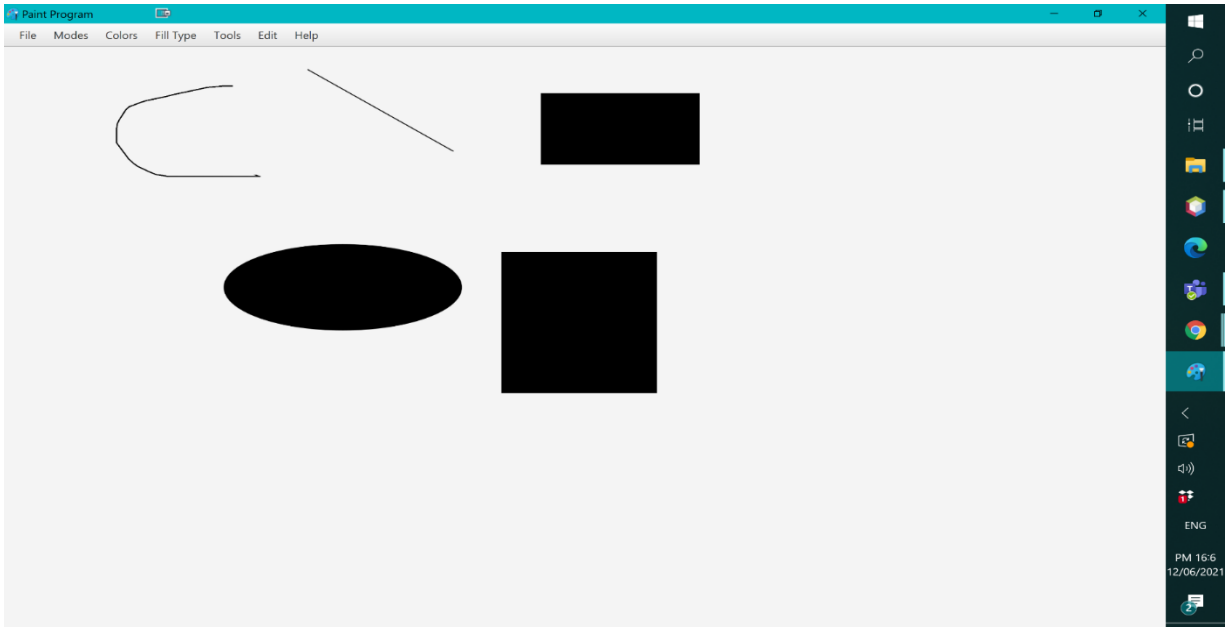
We tried to apply this principle but not fully as we tried to separate the calculations of the positions of different shapes (circles ands squares) as they have different implementation

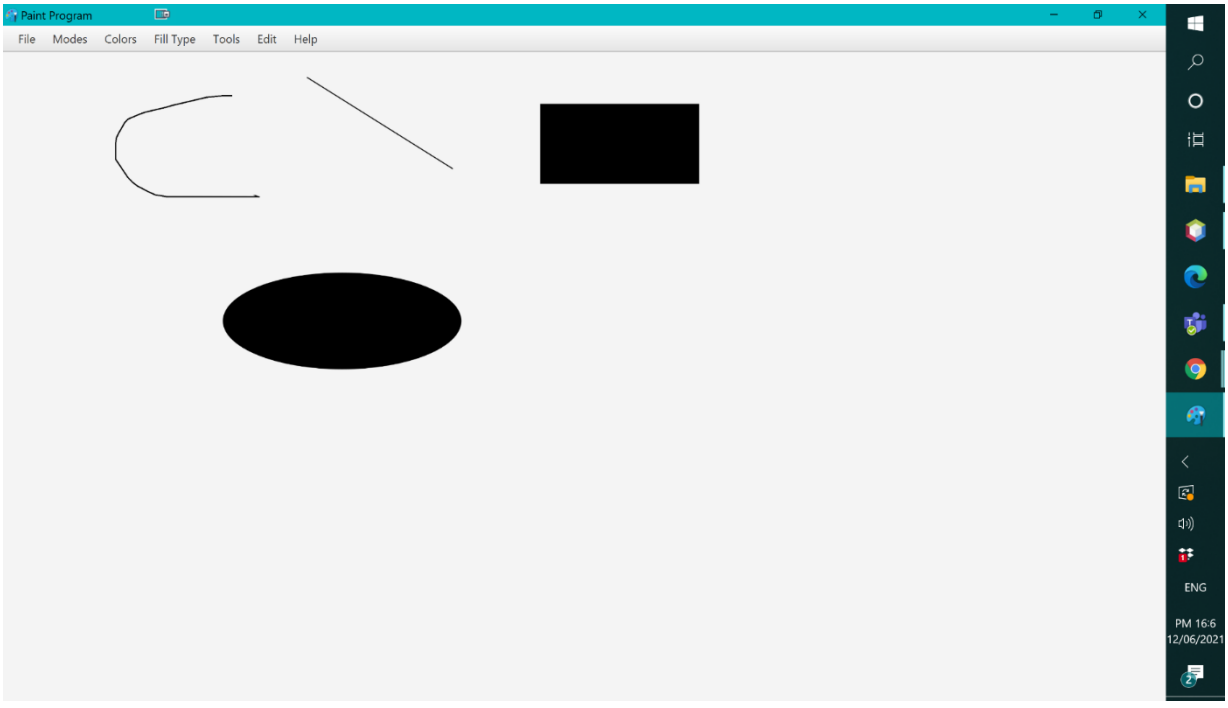
Snapshots of GUI:

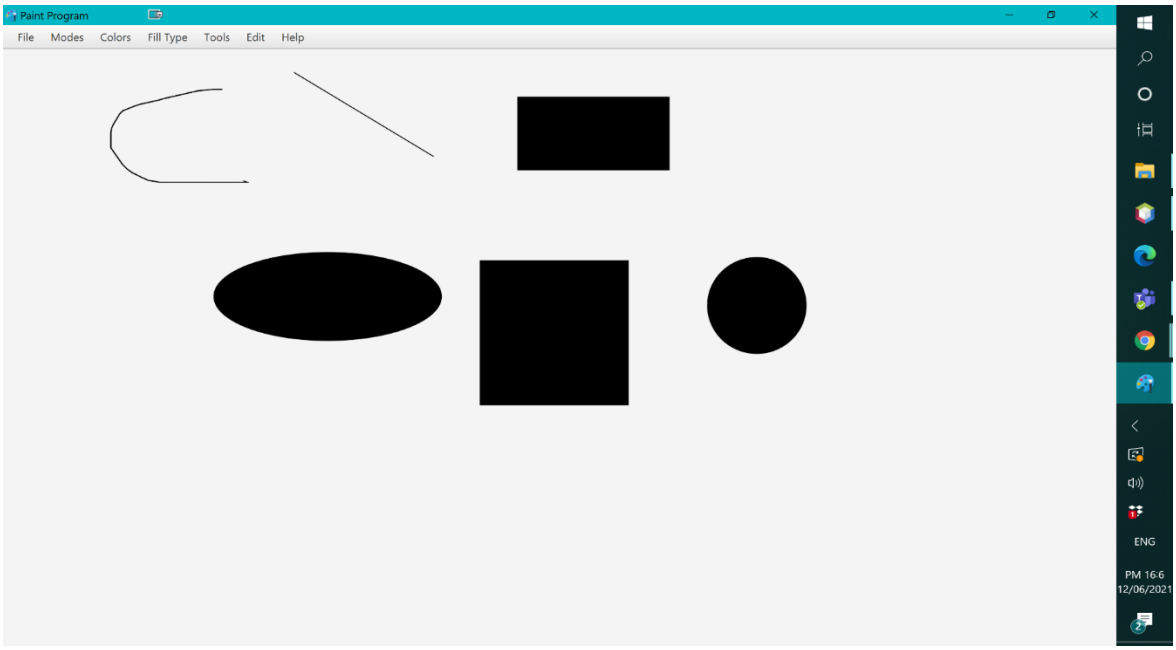


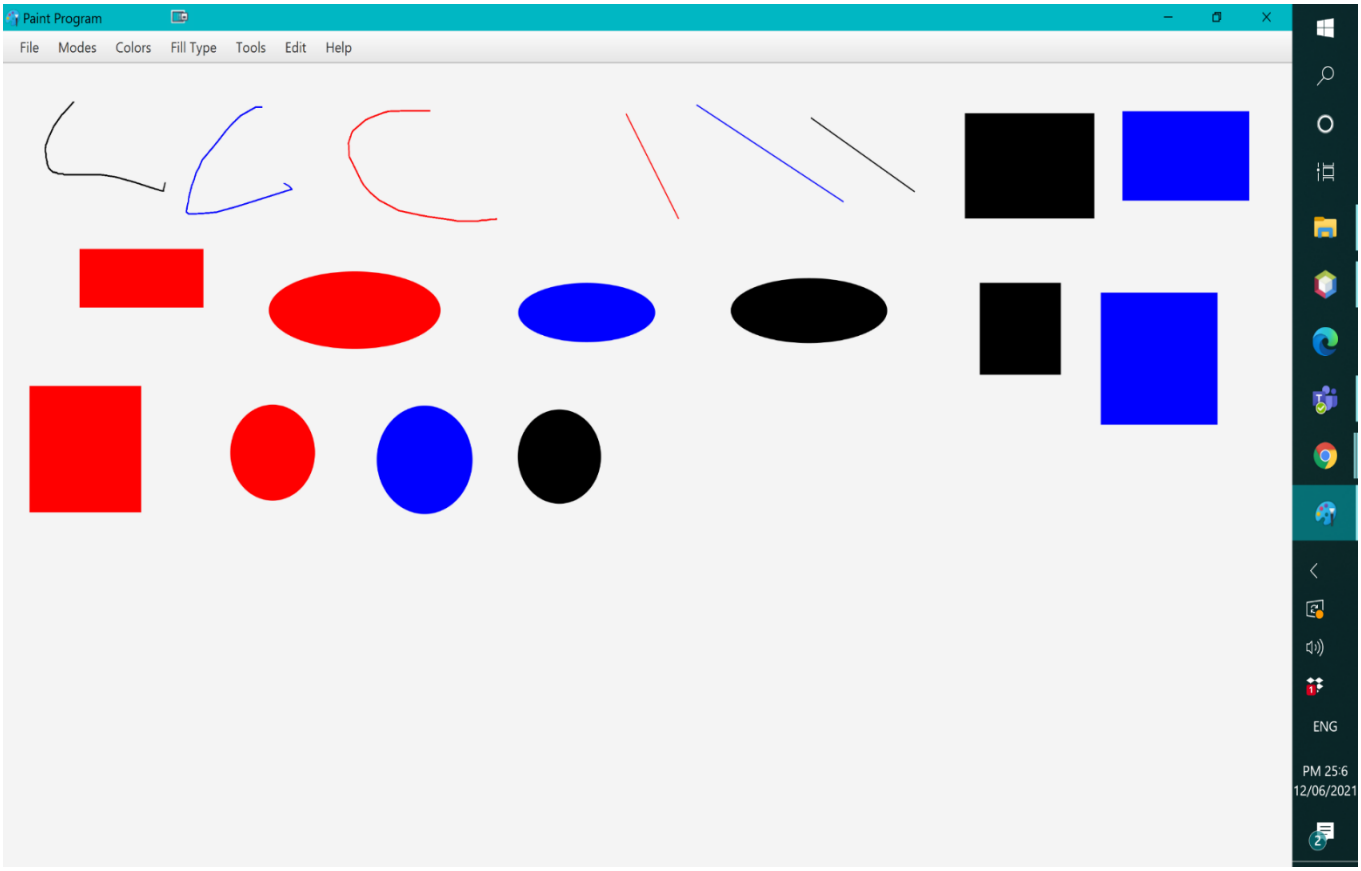

paintProject - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Output - paintProject (run)
Paint Program
File Modes Colors Fill Type Tools Edit Help
Output
paintProject (run)
running...
258:10
INS
ENG
7:10 PM
6/11/2021

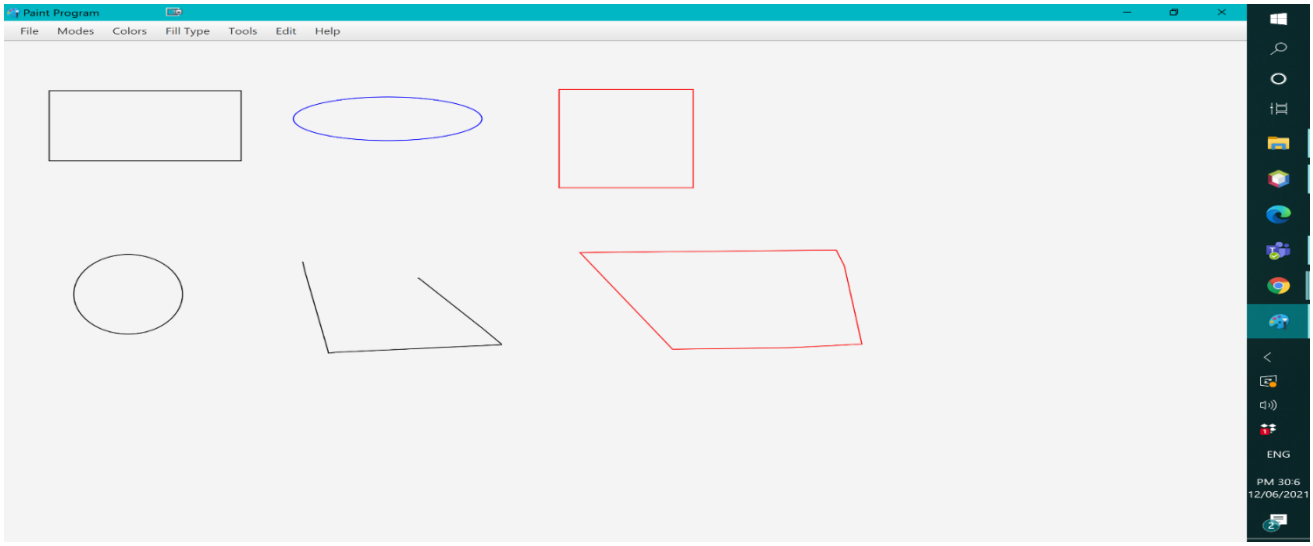

Paint Program
File Modes Colors Fill Type Tools Edit Help
Scribble
Straight Lines
Rectangles
Ellipses
Squares
Circles
Open Polygon
Closed Polygon
ENG
PM 15:6
12/06/2021

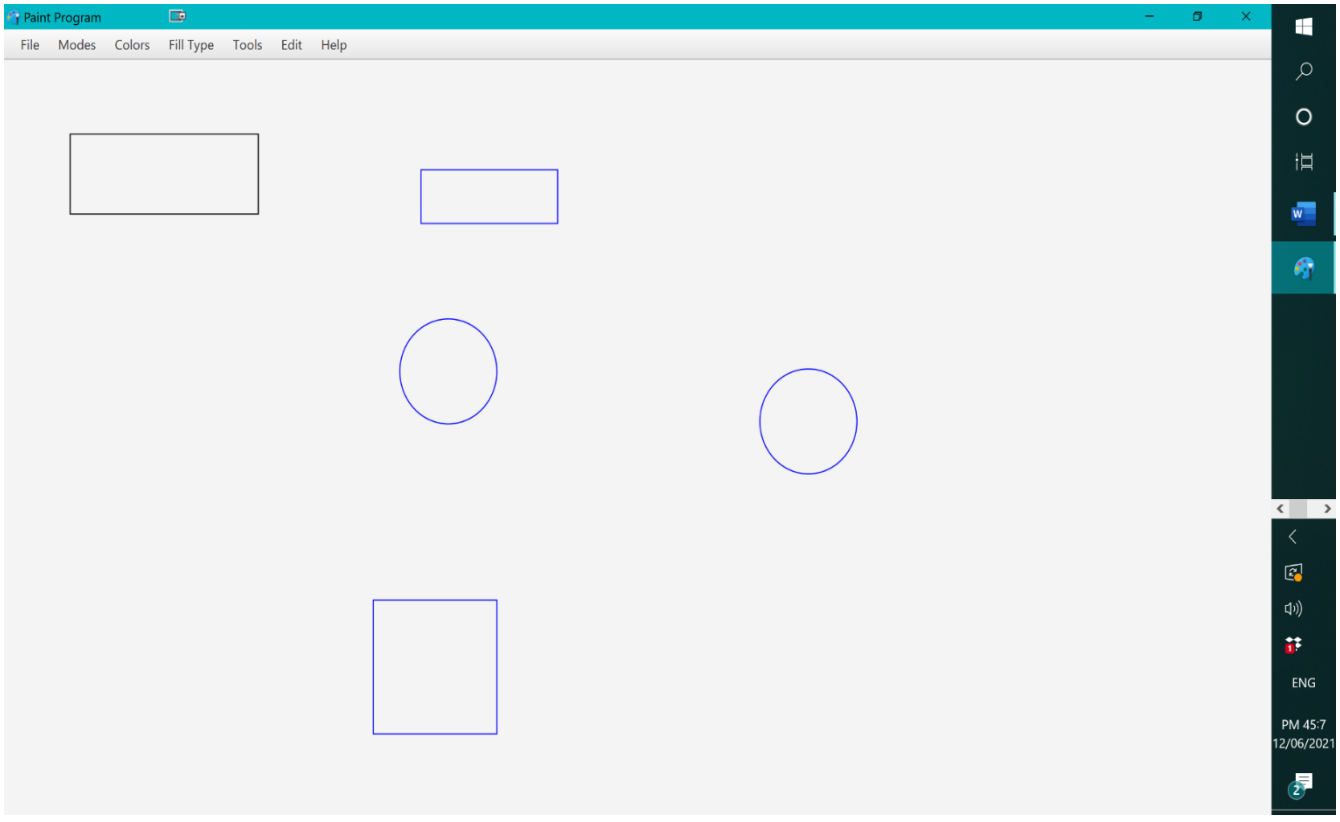

Paint Program
File Modes Colors Fill Type Tools Edit Help
ENG
PM 16:6
12/06/2021


Paint Program
File Modes Colors Fill Type Tools Edit Help
ENG
PM 16:6
12/06/2021


Paint Program
File Modes Colors Fill Type Tools Edit Help
ENG
PM 16:6
12/06/2021


Paint Program
File Modes Colors Fill Type Tools Edit Help
ENG
PM 25:6
12/06/2021


Paint Program
File Modes Colors Fill Type Tools Edit Help
ENG
PM 30:6
12/06/2021


Paint Program
File Modes Colors Fill Type Tools Edit Help
ENG
PM 45:7
12/06/2021