Assignment #3:Paint

Name1: **Yousef Mohamed Hassan Elmedany**

ID1: **20012293**

Name2: **Mohamed Wael Fathy**

ID2: **20011752**

Name3: **Ahmed Samir Said**

ID3: **20010107**

Name4: **Ahmed Khalil Elzainy**

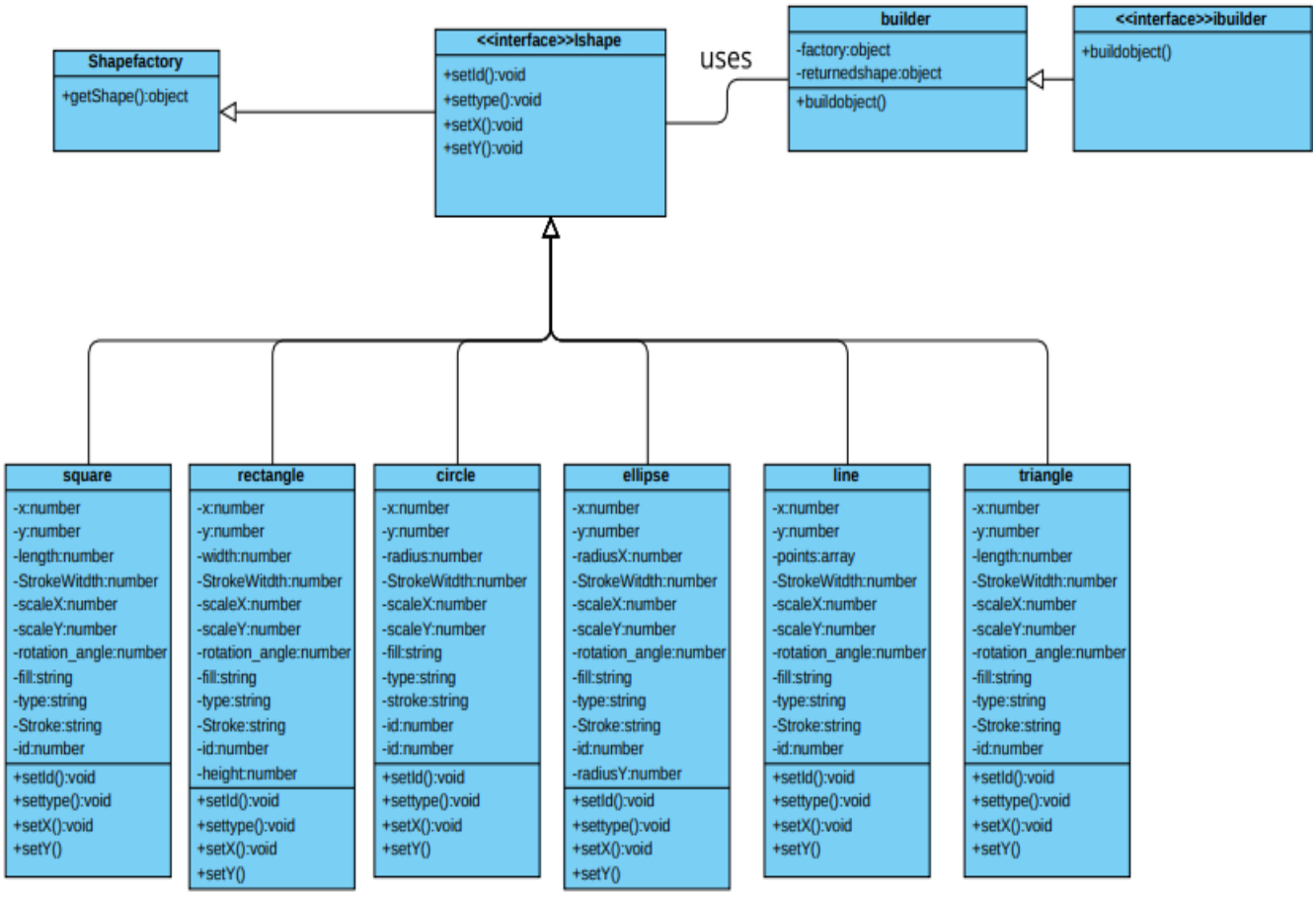
ID4: **20010087**

Steps to Run the Code:

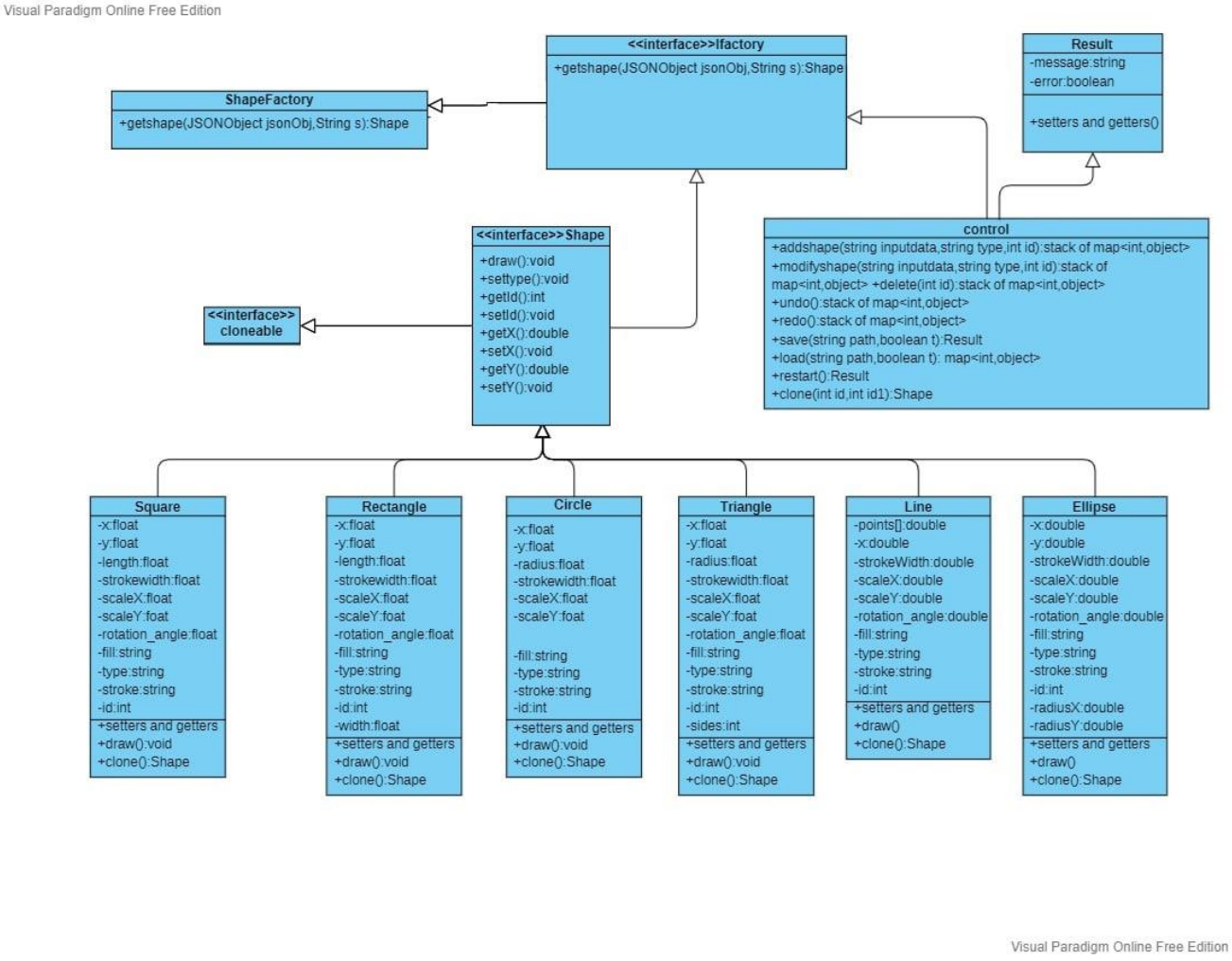
- **Run the IDE and import the folder containing paint files Created by angular for the Frontend implementation.**
- **Switch to the directory containing the project.**
- **Serve the angular project by running “ng serve -o” in the terminal.**
- **Open the browser on the localhost:4200 where the angular live server is listed on.**
- **For the linkage by the backend Code, open another IDE window and import the folder containing the files created by the help of the spring boot.**
- **Run the paint application file that contain the main method for running the whole application.**
- **Backend would be hosted on the localhost:8080, switch off any service hosted on this port or change the port of the backend**

UML:

- **Front-End class diagram**



- **Back-End class diagram**



Design patterns:

- ## Front-end
  - **We used factory design pattern for creation of the shape that make the client class independent for the creation and decision making for the shape type. The type of the shape is passed as a string to the method getshape() and the corresponding shape is created and returned.**

  - **We used builder design pattern to build the shape and give it its attributes to make the client class independent from these complex operations. We just pass the type of the shape and the shape object created by the factory to the method MakeObject() and the attributes of that objects then set.**
- ## Back-end
  - **We use factory design pattern to for creation of the shape that make the client class independent for the creation and decision making for the shape type. When the object received from the front part in addition to the shape type and its id, a factory object is used to create the object by passing the json object received in addition to its type to the getshape method that return the created object.**

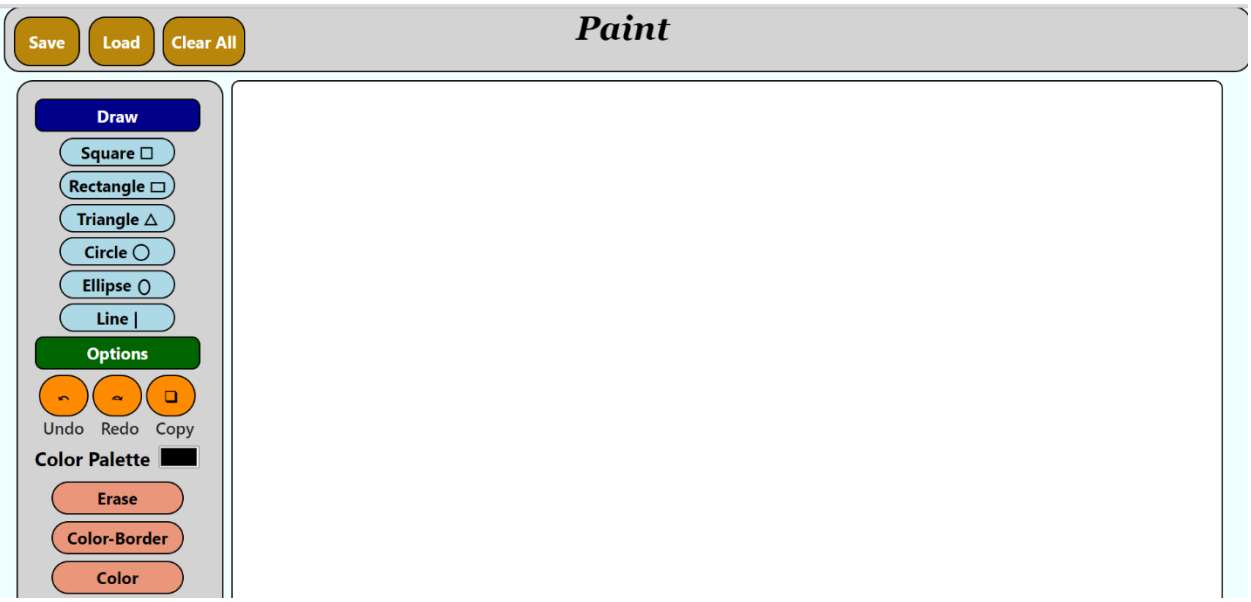
  - **We used prototype design pattern for clone shapes thus the client wouldn't need to make the same complex operation to build this shape. A method clone() that is implemented in the classes of shapes that return a new object shape having the same attributes of the object that is called by.**
  - **We used markable design pattern to mark the Shape interface with cloneable method that make all the shapes able to be cloned.**

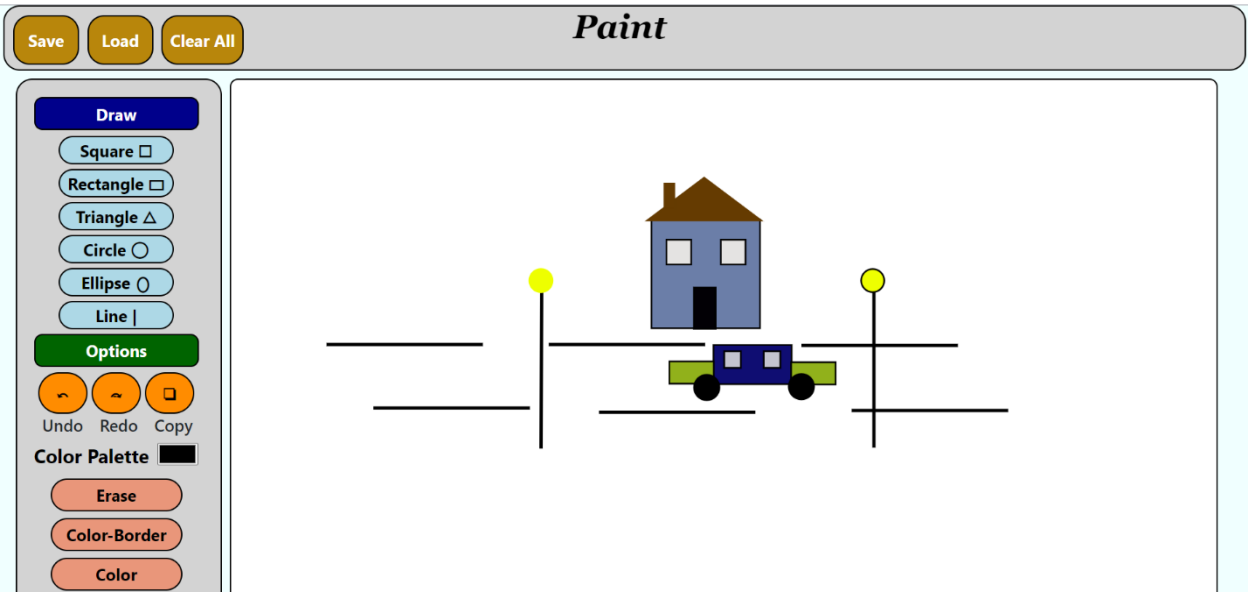
Design decisions:

- ## Front-end
  - **We use konva library to help with draw the shapes which was a library integrated on it function to draw the shapes.**
  - **We make a counter of ids to give each shape unique id.**
  - **For creation and modification of the shape we send the shape, its id and its type to the backend to be stored.**
  - **For undo and redo we receive layer of shapes (map) and redraw it.**
  - **For saving the drawn shapes a popup window appears so user could pass the name, file path and choose the type of the file json or xml.**
  - **For load a file a popup window appears, and the user pass the path of the file, but the user can't undo or redo on loaded shapes.**
- ## Back-end
  - **We decide to save the current state for the stage on a map of integer -that represent the id- and object -that represent the shape-so it would be the current layer of the application.**
  - **We decide to use stack of maps that represent all layers of the application from the beginning to the end, which this decision was our solution for undo and redo all steps of the app.**
  - **We save the Ids of the shapes on undo and redo stacks of integers to control the flow of the application.**
  - **To avoid the problem of reference we decided to create new object and clone the desired data into it.**
  - **For undo we pop the first layer of the undo stack and clone it then push it onto redo stack and assign it to the current map.**
  - **For redo we pop the first layer of the redo stack and clone it then push it again onto undo stack.**

Design Decisions:

UI:





User Manual:

- **To create a new shape, press on the button of the required shape form the Draw section then place the cursor anywhere on the whiteboard to draw it.**
- **To change the color of a shape, select the color first from the color palette then press "Color" button and go to the shape you want to change its color and point to it.**
- **To change the border color of a certain shape, select the color from the color palette then press on "Color-Border" button then go to the shape you want to change its border and point to it.**
- **To Resize any shape, go to the shape and point to it by the cursor to select it and then resize by grabbing its corners.**
- **To copy a shape, press on "Copy" button an then go and select the shape you want to copy and click on it.**
- **To Erase any thing on the whiteboard, press "Erase" button and then go to the thing you want to erase and click on it by the cursor.**
- **To undo an action you performed, just click on the "Undo" button.**
- **To redo an action after undo just click on the "Redo" button.**
- **To reset the whole whiteboard and clear everything, go to the "Clear All" button in the header and press it.**
- **To save a work-file, go to the "Save" button in the header and click on it, a pop-up dialogue will be shown. You will be required to enter the name of the file and its path and select either to save it in .json or .xml extension. After filling the requirements and click on done button.**
- **To load a file previously saved from PC, go to the "Load" button in the header and click on it, a pop-up dialogue will be shown. You will be asked to enter the path of the file you want to load and after filling it press on "Enter" button and then the file will be loaded.**