Alexandria University
Faculty of Engineering
Computer & Systems Engineering
CS233: Computer Organization

Paint Lap oop

Student Name	Student ID
Karim Fathy Abd-Elaziz	20011116
Ahmed Mohamed Hassan	20010174
Mohamed Ahmed Ali Ryad	20011457
FARIS MOHAMED ANWAR	20011065

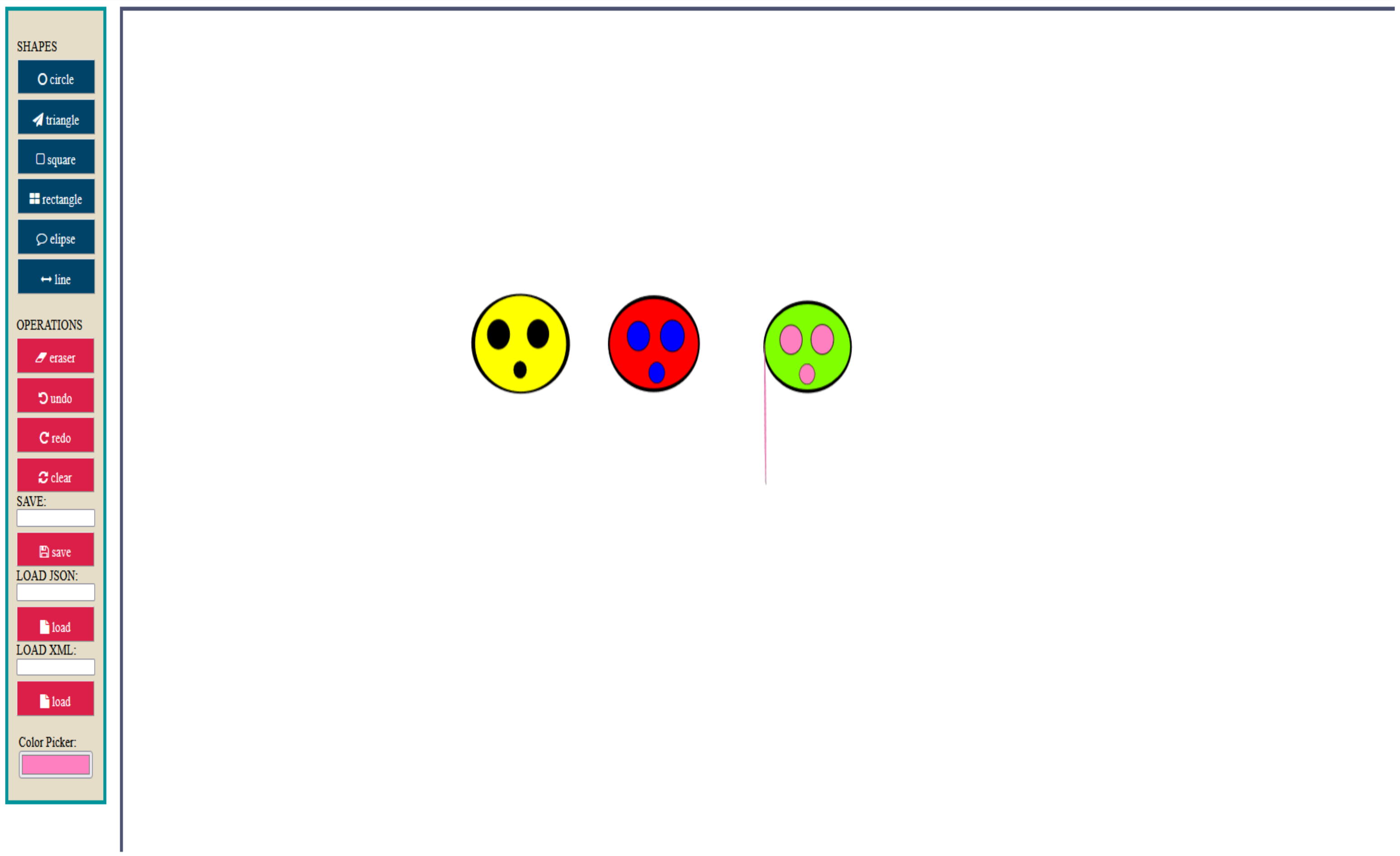
Part 1: Geometric Shapes Data Model:

Tasks

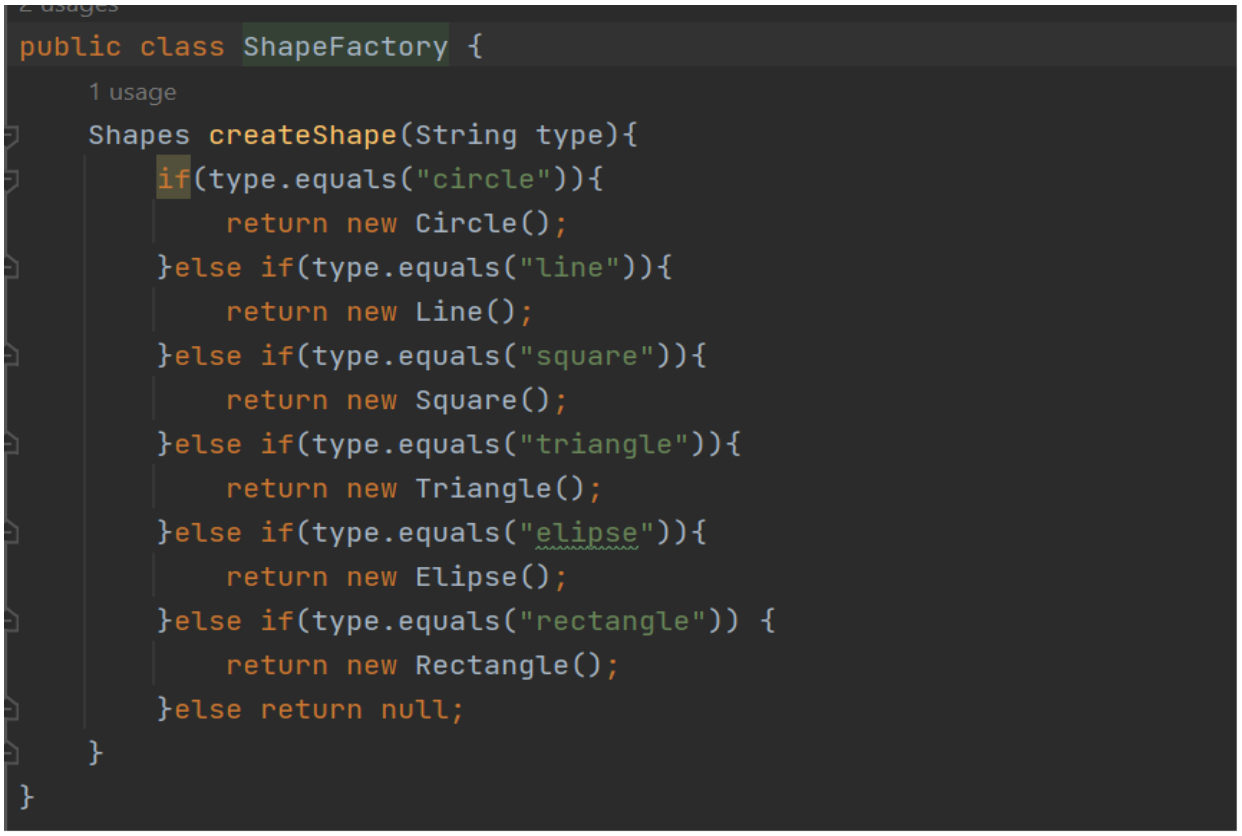
1. Design an object-oriented model that covers the following geometric shapes: Line Segment, Circle, Ellipse, Triangle, Rectangle and Square.

2. Draw a UML Class diagram that represents your model, showing all the classes, attributes and methods.

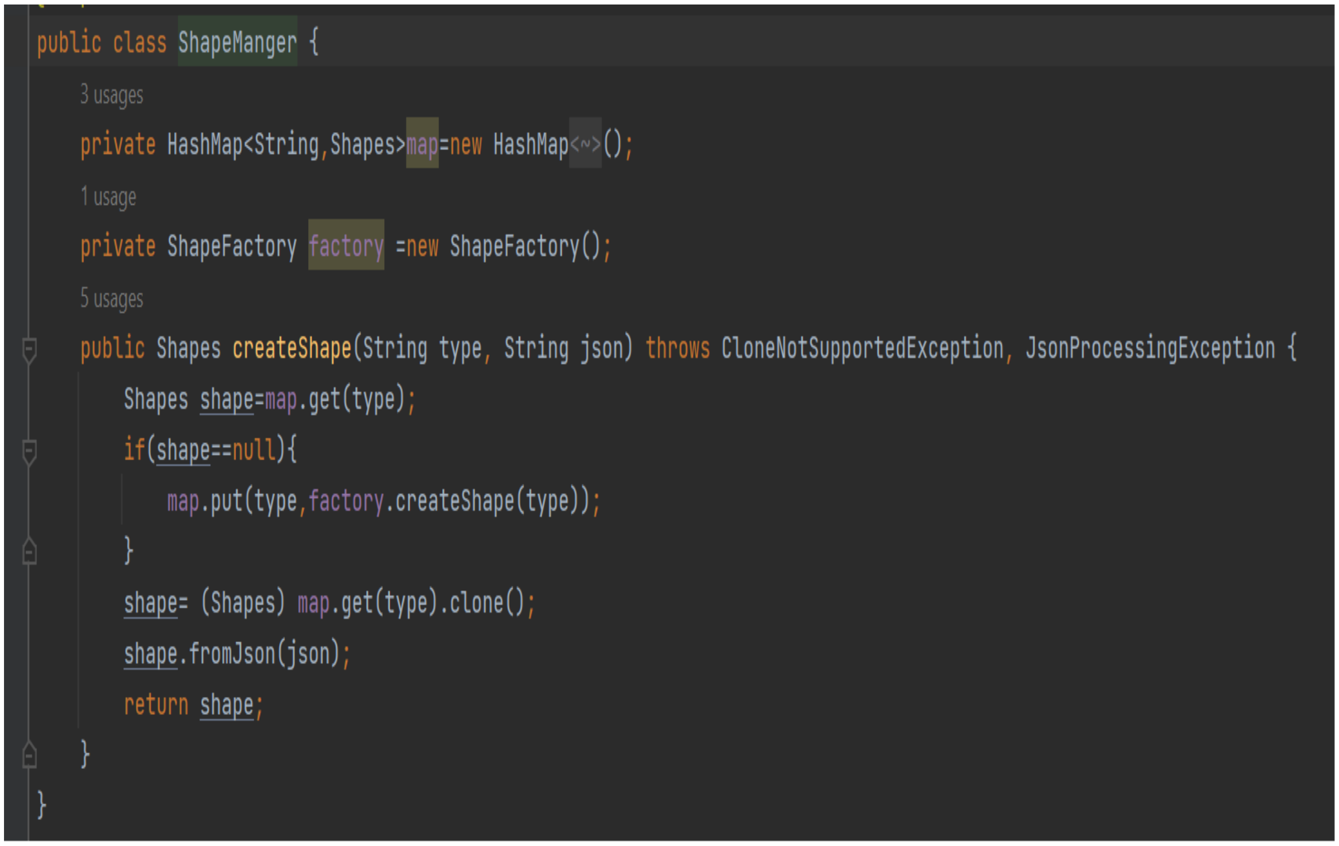
3. Apply the concepts of inheritance and polymorphism to your design.



First : we apply Factory Design pattern.

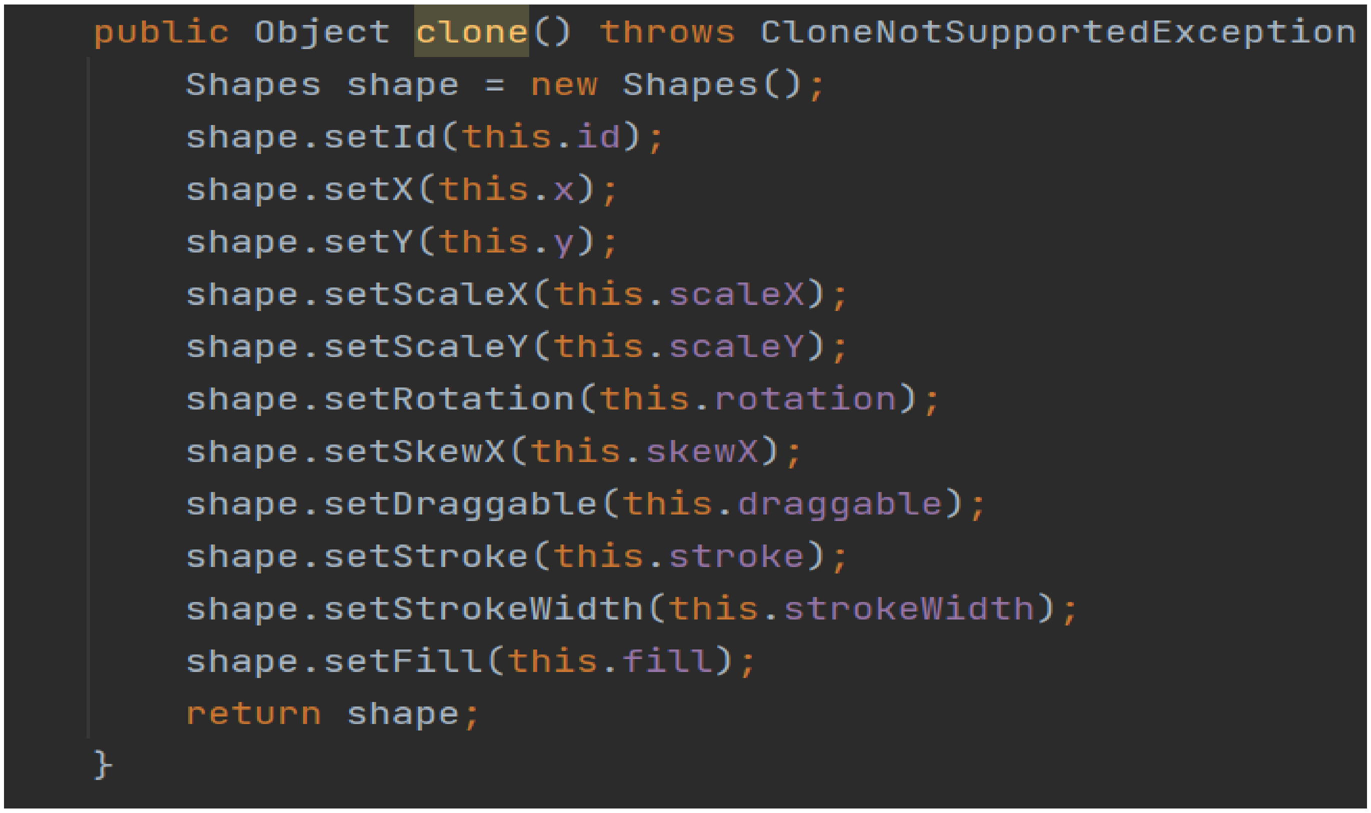


```
2 usages
public class ShapeFactory {
    1 usage
    Shapes createShape(String type){
        if(type.equals("circle")){
            return new Circle();
        }else if(type.equals("line")){
            return new Line();
        }else if(type.equals("square")){
            return new Square();
        }else if(type.equals("triangle")){
            return new Triangle();
        }else if(type.equals("elipse")){
            return new Elipse();
        }else if(type.equals("rectangle")) {
            return new Rectangle();
        }else return null;
    }
}
```



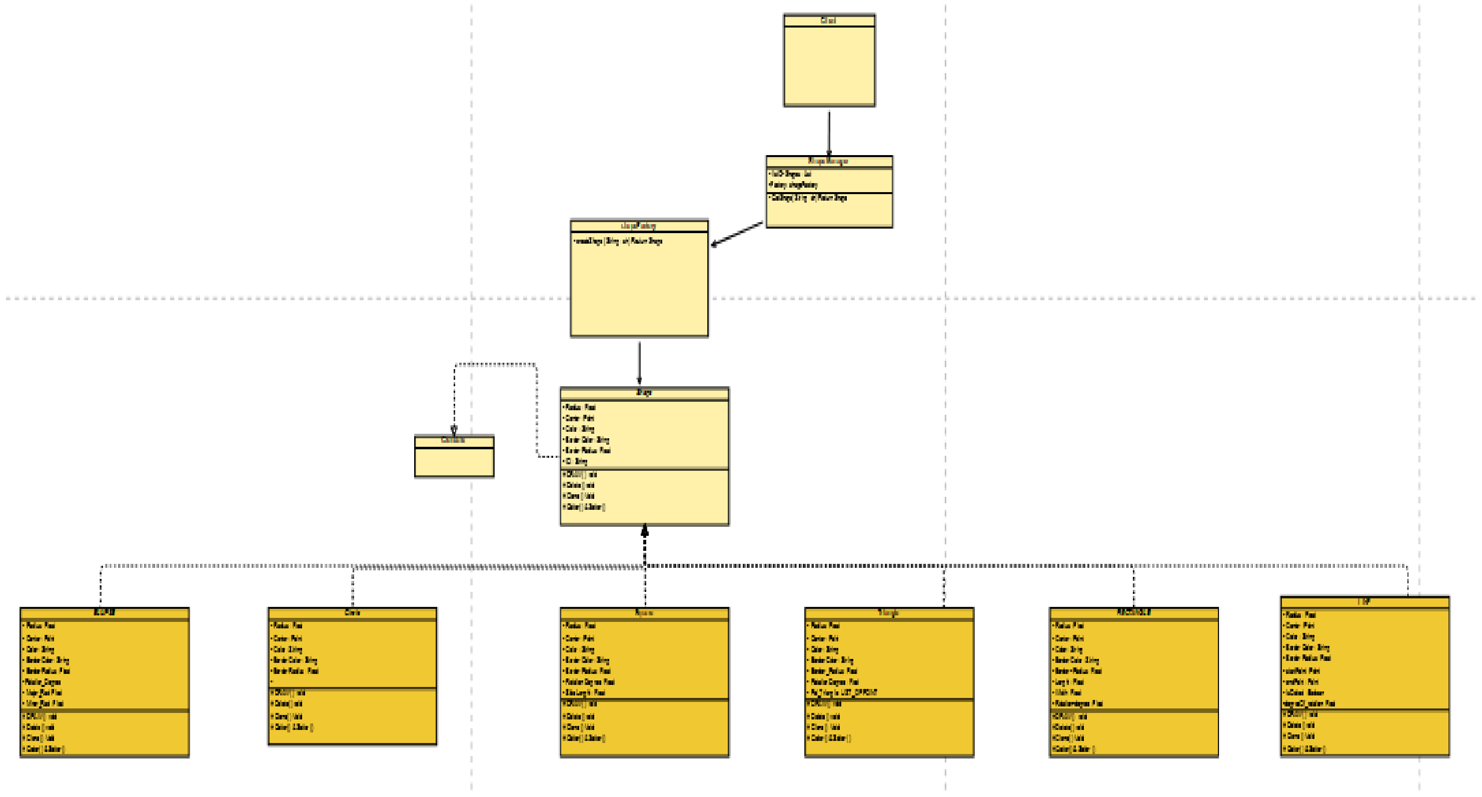
```java
public class ShapeManger {
    3 usages
    private HashMap<String,Shapes>map=new HashMap<~>();
    1 usage
    private ShapeFactory factory =new ShapeFactory();
    5 usages
    public Shapes createShape(String type, String json) throws CloneNotSupportedException, JsonProcessingException {
        Shapes shape=map.get(type);
        if(shape==null){
            map.put(type,factory.createShape(type));
        }
        shape= (Shapes) map.get(type).clone();
        shape.fromJson(json);
        return shape;
    }
}
```

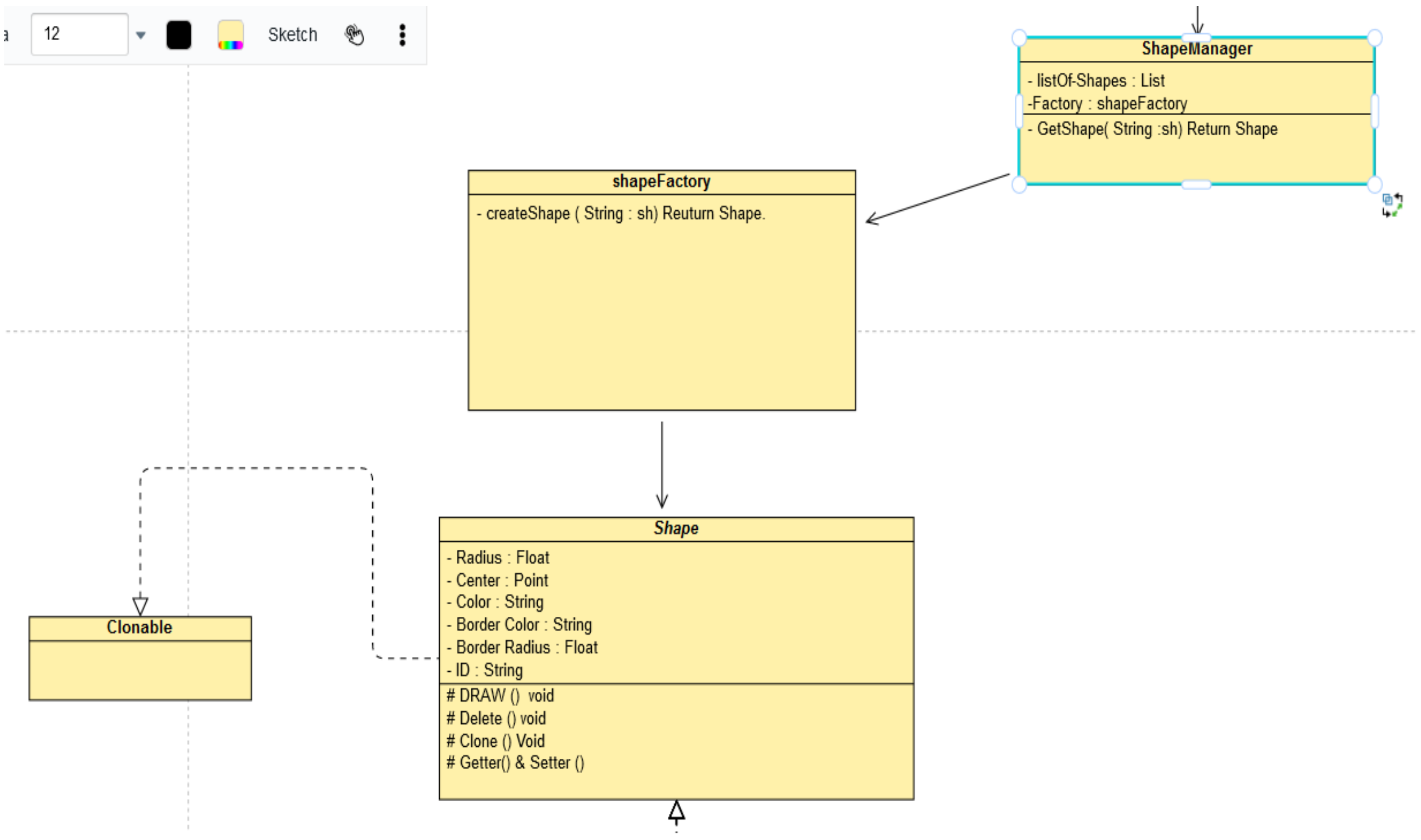
We used Prototype Design pattern:



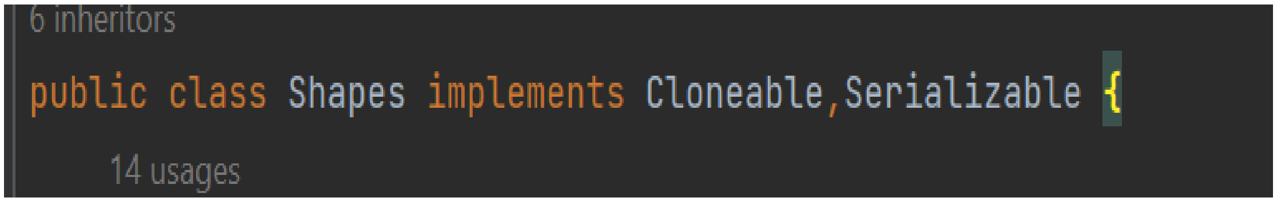
```java
    public Object clone() throws CloneNotSupportedException
        Shapes shape = new Shapes();
        shape.setId(this.id);
        shape.setX(this.x);
        shape.setY(this.y);
        shape.setScaleX(this.scaleX);
        shape.setScaleY(this.scaleY);
        shape.setRotation(this.rotation);
        shape.setSkewX(this.skewX);
        shape.setDraggable(this.draggable);
        shape.setStroke(this.stroke);
        shape.setStrokeWidth(this.strokeWidth);
        shape.setFill(this.fill);
        return shape;
    }
```

As required UML for shapes

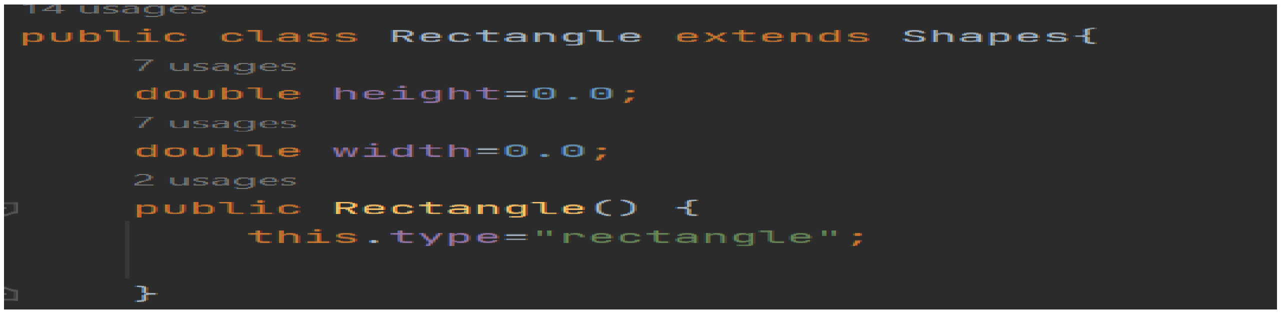




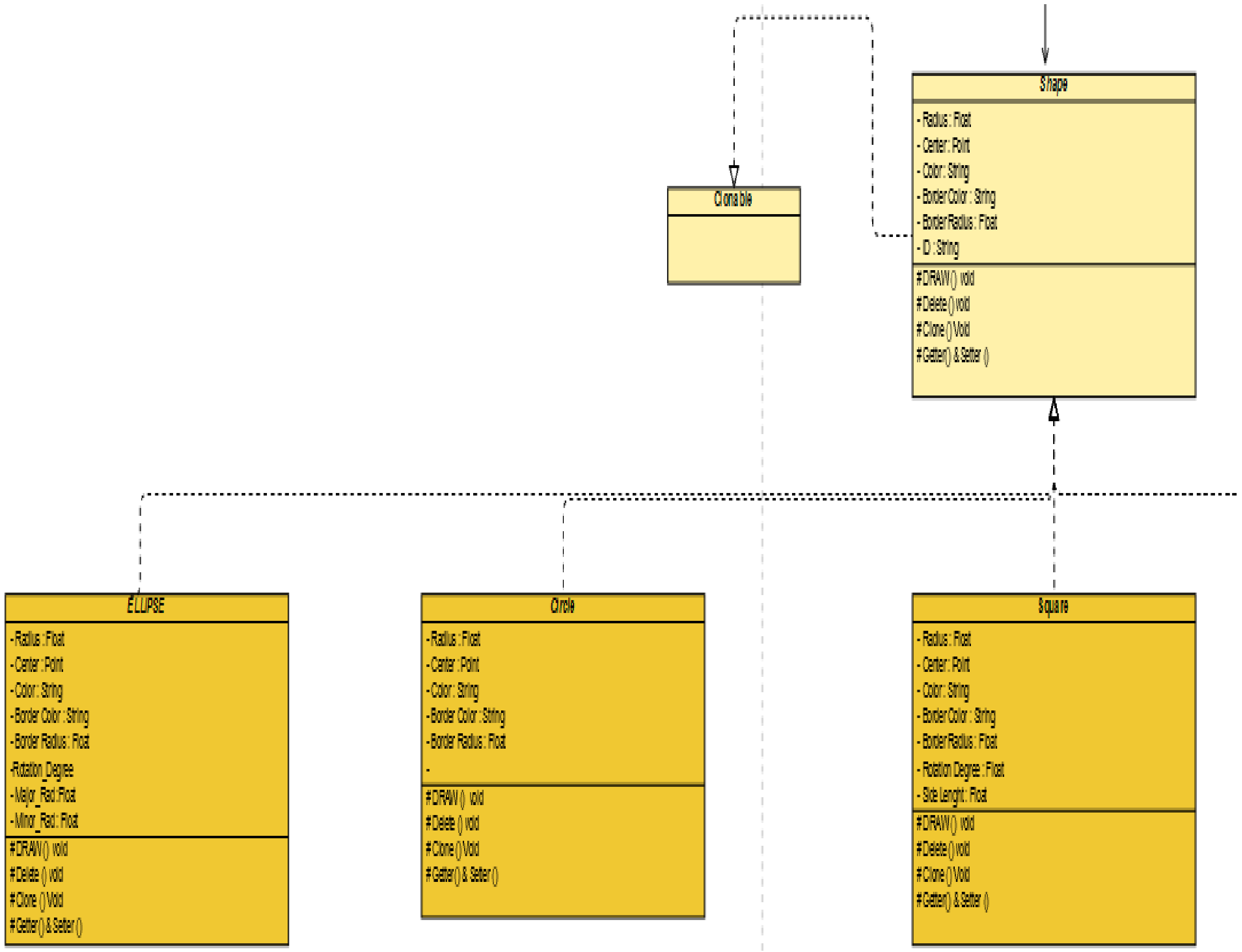
We Apply the concepts of inheritance and polymorphism to your design:



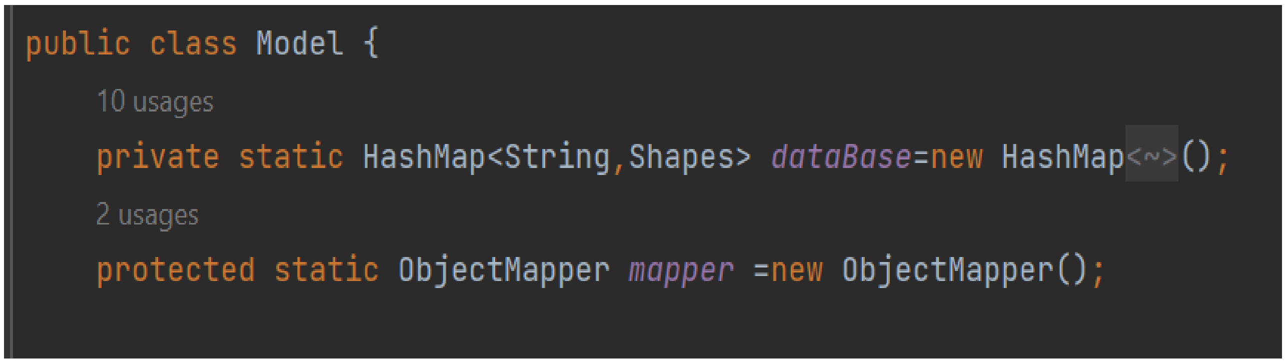
```
6 inheritors
public class Shapes implements Cloneable,Serializable {
    14 usages
```



```
14 usages
public class Rectangle extends Shapes{
        7 usages
        double height=0.0;
        7 usages
        double width=0.0;
        2 usages
        public Rectangle() {
            this.type="rectangle";

        }
```



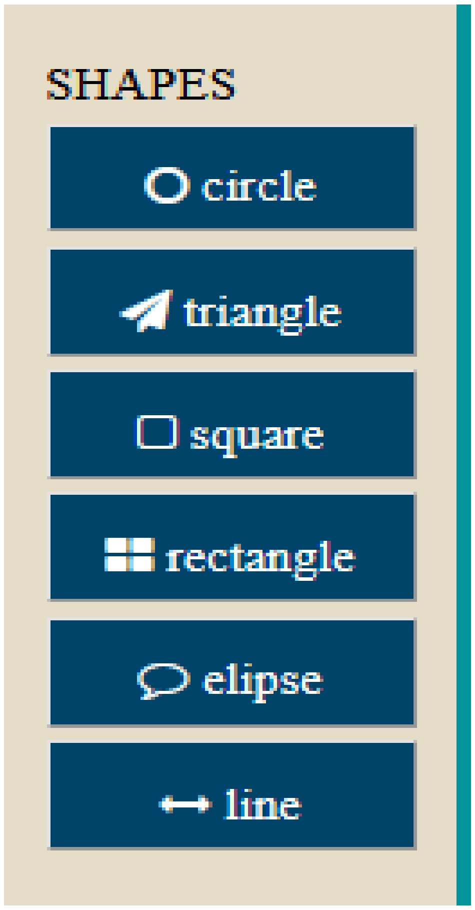
We Have Model in it We have a Global Map which called Database:

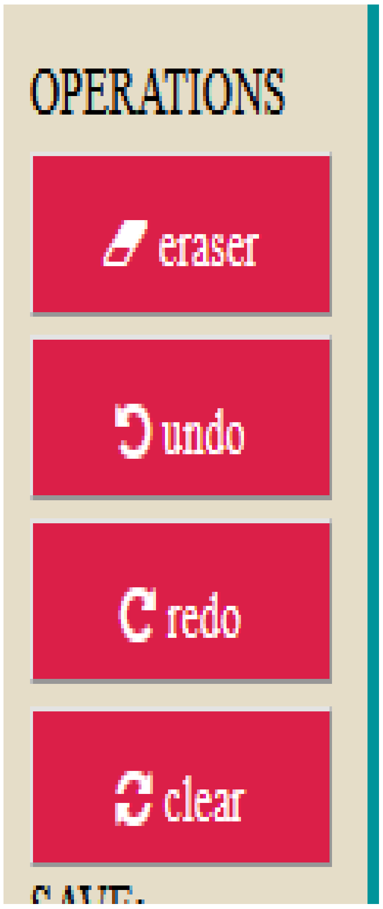


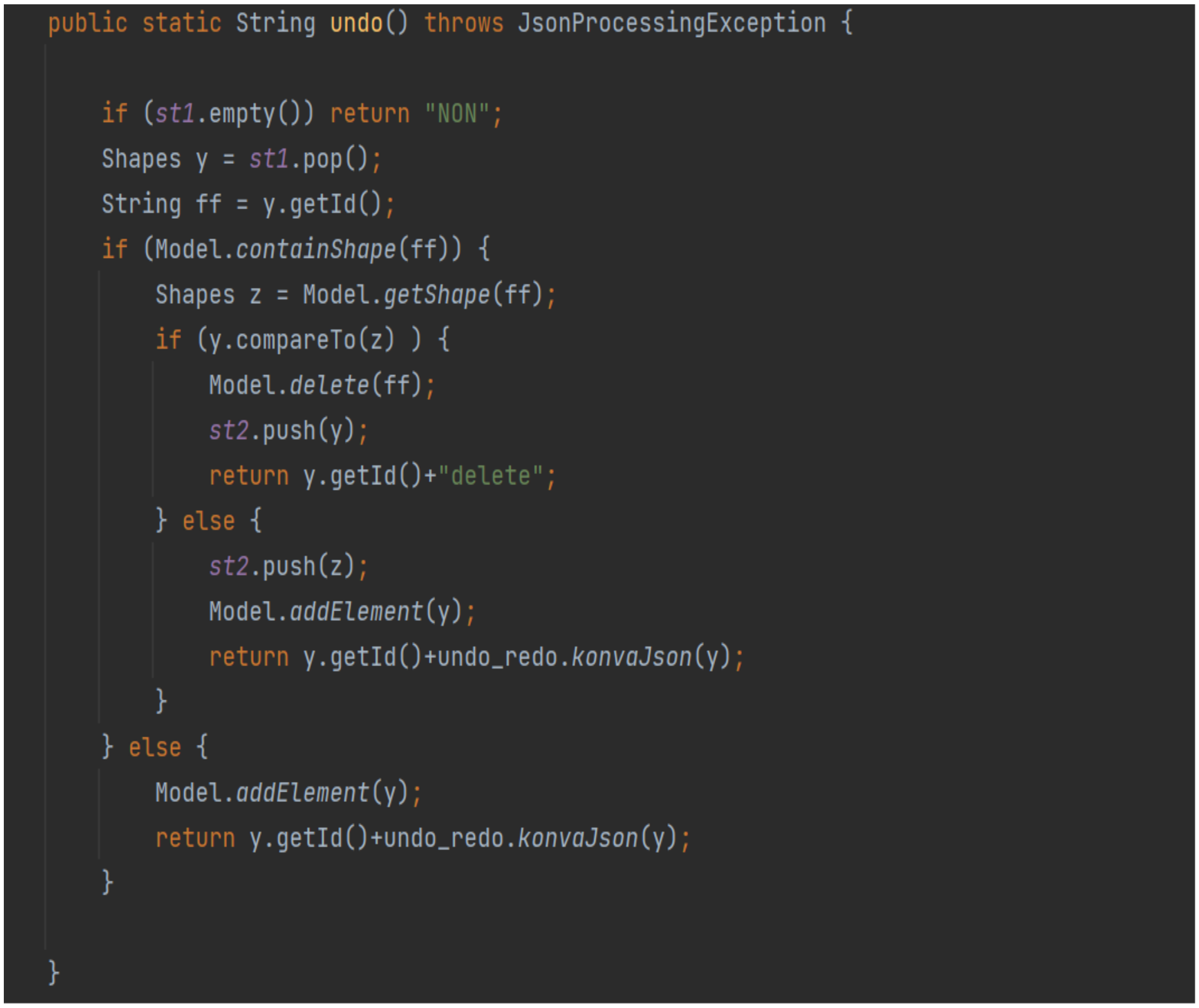
```
public class Model {
    10 usages
    private static HashMap<String,Shapes> dataBase=new HashMap<~>();
    2 usages
    protected static ObjectMapper mapper =new ObjectMapper();
```

We used This map to be center of program .

We implement a GUI that allows the following functionalities for the user on all The shapes : Draw, Color, Resize, Move, and Delete,Roatate.
Implement your application such that it would allow the user to undo or redo any action performed.

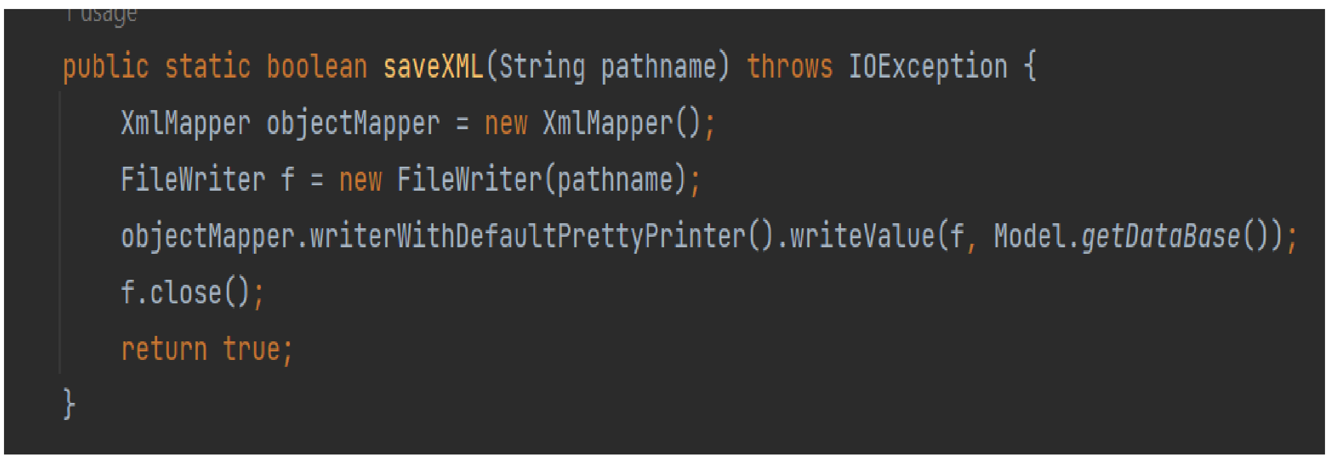






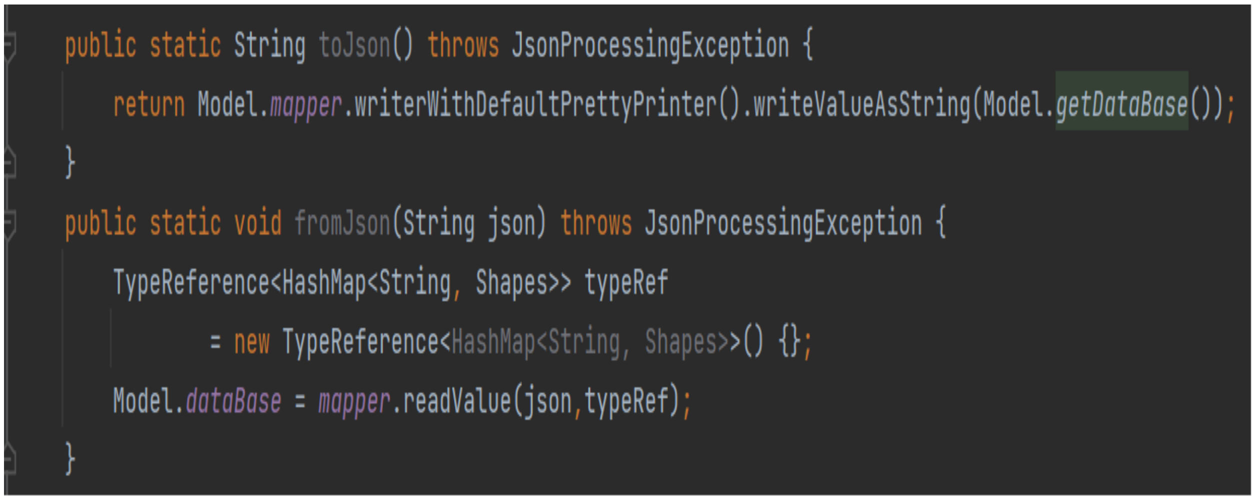
```java
public static String undo() throws JsonProcessingException {

    if (st1.empty()) return "NON";
    Shapes y = st1.pop();
    String ff = y.getId();
    if (Model.containShape(ff)) {
        Shapes z = Model.getShape(ff);
        if (y.compareTo(z) ) {
            Model.delete(ff);
            st2.push(y);
            return y.getId()+"delete";
        } else {
            st2.push(z);
            Model.addElement(y);
            return y.getId()+undo_redo.konvaJson(y);
        }
    } else {
        Model.addElement(y);
        return y.getId()+undo_redo.konvaJson(y);
    }

}
```

WE -Provide an option in UI to save the drawing in XML (encoding: ISO-8859-1) and JSON
Provide an option to load previously saved drawings and modify the shapes.
 Users must choose where to save the File save And The path to laod.

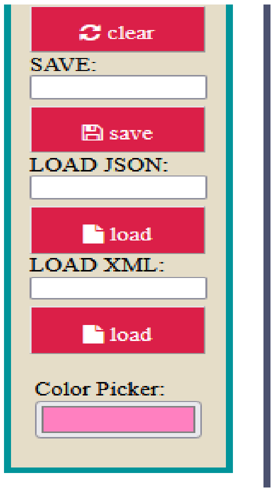


```java
1 usage
public static boolean saveXML(String pathname) throws IOException {
    XmlMapper objectMapper = new XmlMapper();
    FileWriter f = new FileWriter(pathname);
    objectMapper.writerWithDefaultPrettyPrinter().writeValue(f, Model.getDataBase());
    f.close();
    return true;
}
```

We implement two fun to convert from obj to json and oppisite

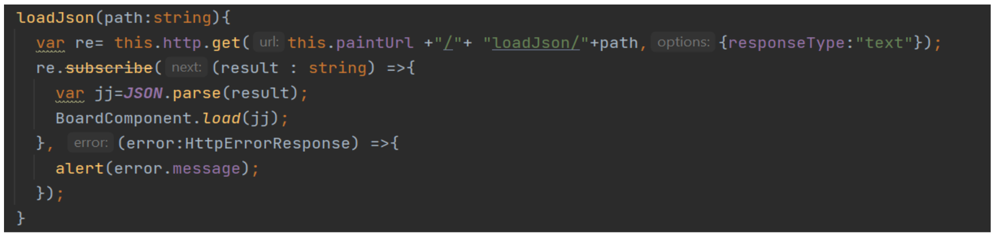


```java
public static String toJson() throws JsonProcessingException {
    return Model.mapper.writerWithDefaultPrettyPrinter().writeValueAsString(Model.getDataBase());
}
public static void fromJson(String json) throws JsonProcessingException {
    TypeReference<HashMap<String, Shapes>> typeRef
            = new TypeReference<HashMap<String, Shapes>>() {};
    Model.dataBase = mapper.readValue(json,typeRef);
}
```



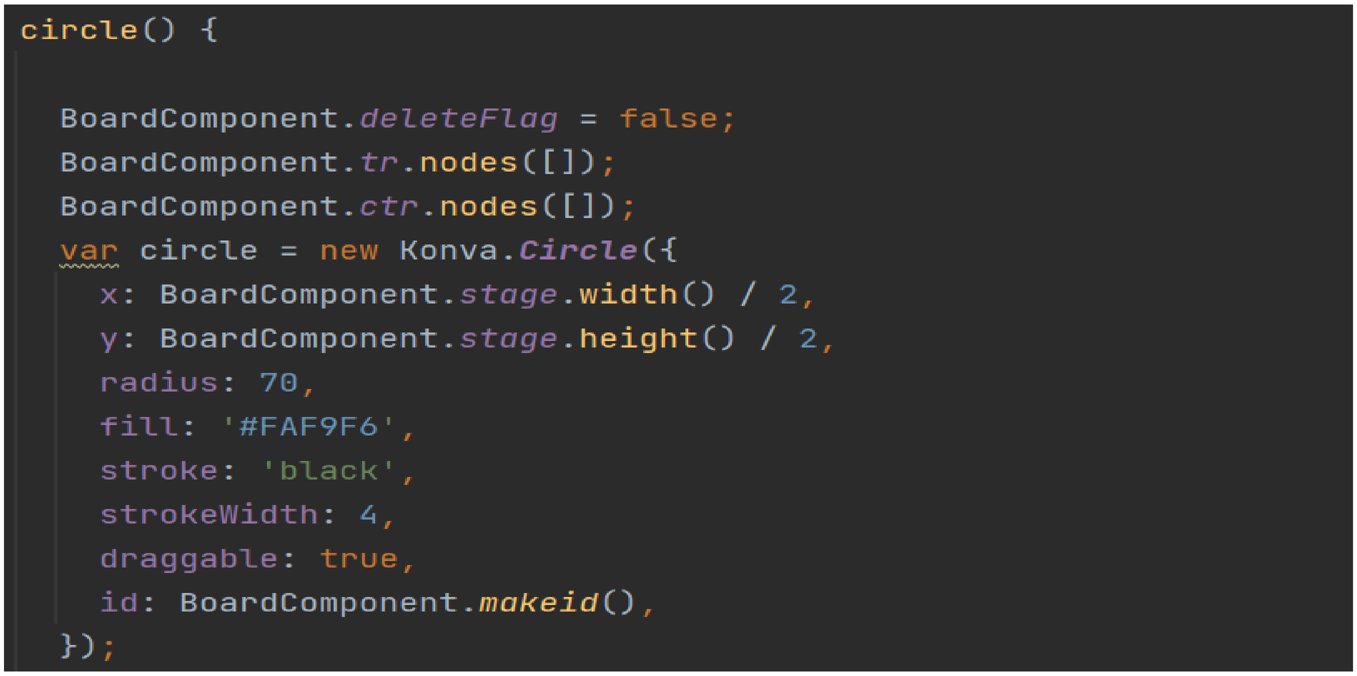
User choose path To save Data of paint and when he want to

Load user has 2 choice Load Xml ,Load Json



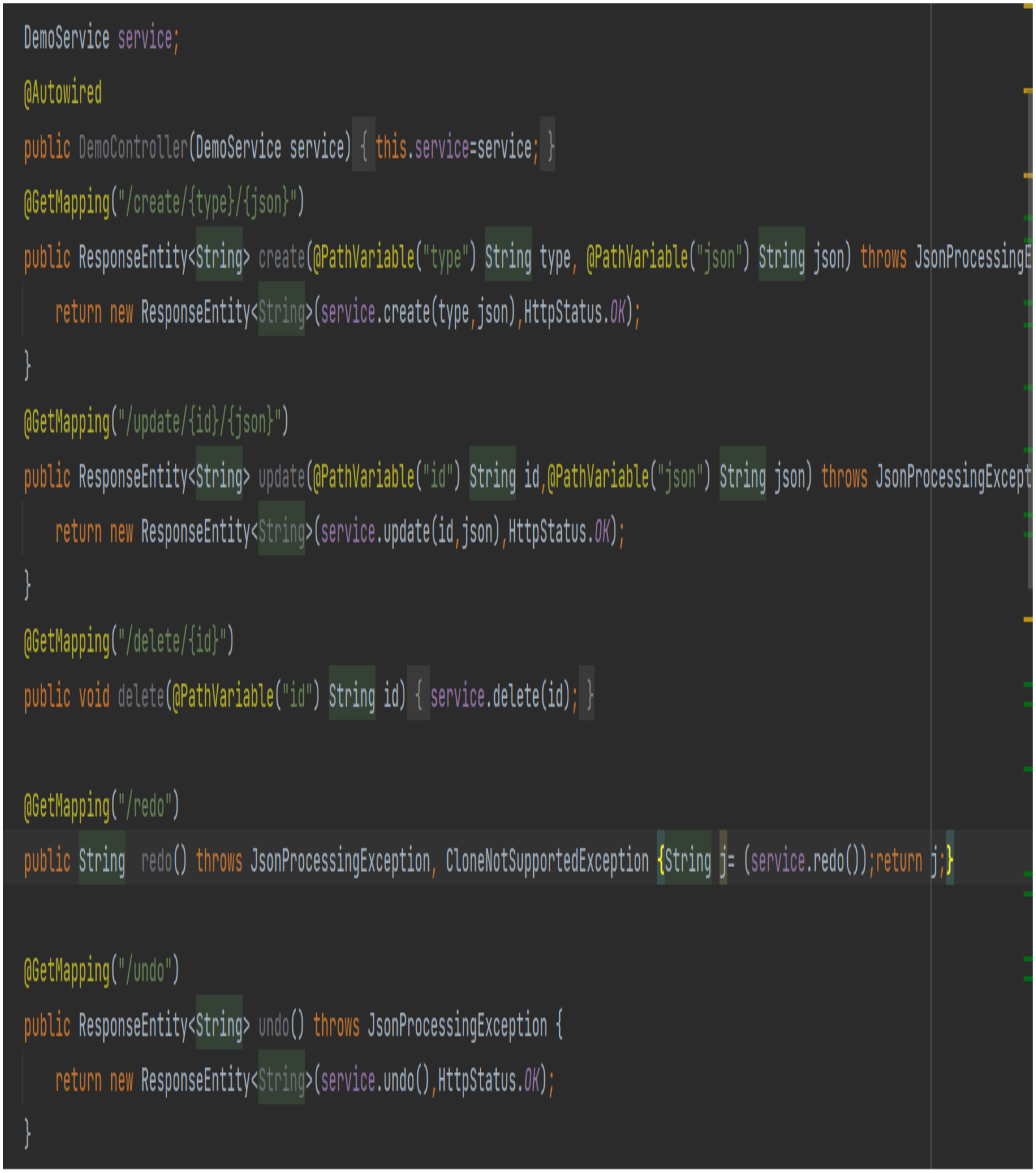
```typescript
loadJson(path:string){
  var re= this.http.get( url: this.paintUrl +"/"+ "loadJson/"+path, options: {responseType:"text"});
  re.subscribe( next: (result : string) =>{
    var jj=JSON.parse(result);
    BoardComponent.load(jj);
  }, error: (error:HttpErrorResponse) =>{
    alert(error.message);
  });
}
```

We used Conva Library to Draw :



```typescript
circle() {

  BoardComponent.deleteFlag = false;
  BoardComponent.tr.nodes([]);
  BoardComponent.ctr.nodes([]);
  var circle = new Konva.Circle({
    x: BoardComponent.stage.width() / 2,
    y: BoardComponent.stage.height() / 2,
    radius: 70,
    fill: '#FAF9F6',
    stroke: 'black',
    strokeWidth: 4,
    draggable: true,
    id: BoardComponent.makeid(),
  });
```

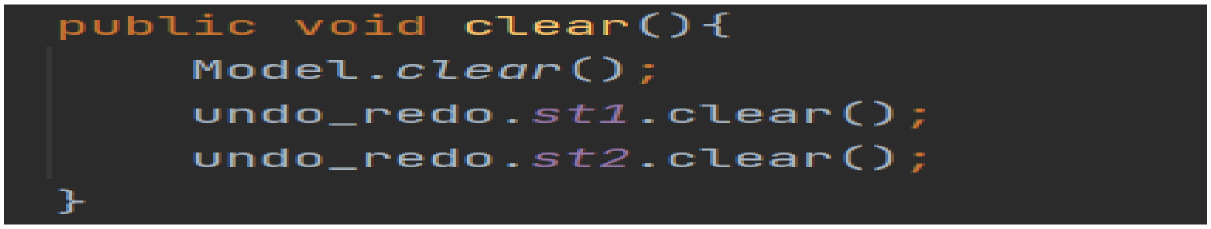
We Connect Between Front and Back Using this fun:



```
DemoService service;
@Autowired
public DemoController(DemoService service) { this.service=service; }
@GetMapping("/create/{type}/{json}")
public ResponseEntity<String> create(@PathVariable("type") String type, @PathVariable("json") String json) throws JsonProcessingE
    return new ResponseEntity<String>(service.create(type,json),HttpStatus.OK);
}
@GetMapping("/update/{id}/{json}")
public ResponseEntity<String> update(@PathVariable("id") String id,@PathVariable("json") String json) throws JsonProcessingExcept
    return new ResponseEntity<String>(service.update(id,json),HttpStatus.OK);
}
@GetMapping("/delete/{id}")
public void delete(@PathVariable("id") String id) { service.delete(id); }


@GetMapping("/redo")
public String  redo() throws JsonProcessingException, CloneNotSupportedException {String j= (service.redo());return j;}


@GetMapping("/undo")
public ResponseEntity<String> undo() throws JsonProcessingException {
    return new ResponseEntity<String>(service.undo(),HttpStatus.OK);
}
```

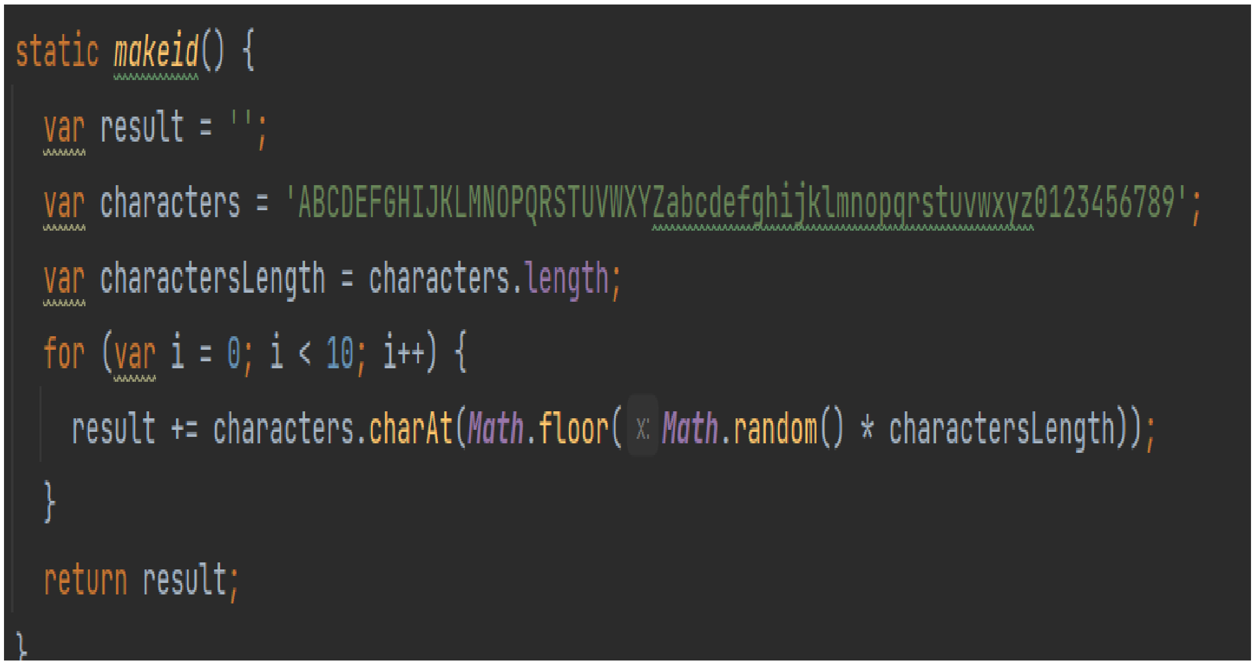
Summary :

We Used Conva To draw in front And we use connection func to send to Back to Update Database and Stack of undo and redo for every operation in front we did that when user want to make undo or redo we sent to stack and update all attributes of obj and Destroy old obj and Create new obj when we refresh or make clear() we clear stack and map



```
public void clear(){
        Model.clear();
        undo_redo.st1.clear();
        undo_redo.st2.clear();
}
```

We make id for every obj in front using func to random string from 10 char



```
static makeid() {
  var result = '';
  var characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
  var charactersLength = characters.length;
  for (var i = 0; i < 10; i++) {
    result += characters.charAt(Math.floor( x: Math.random() * charactersLength));
  }
  return result;
}
```

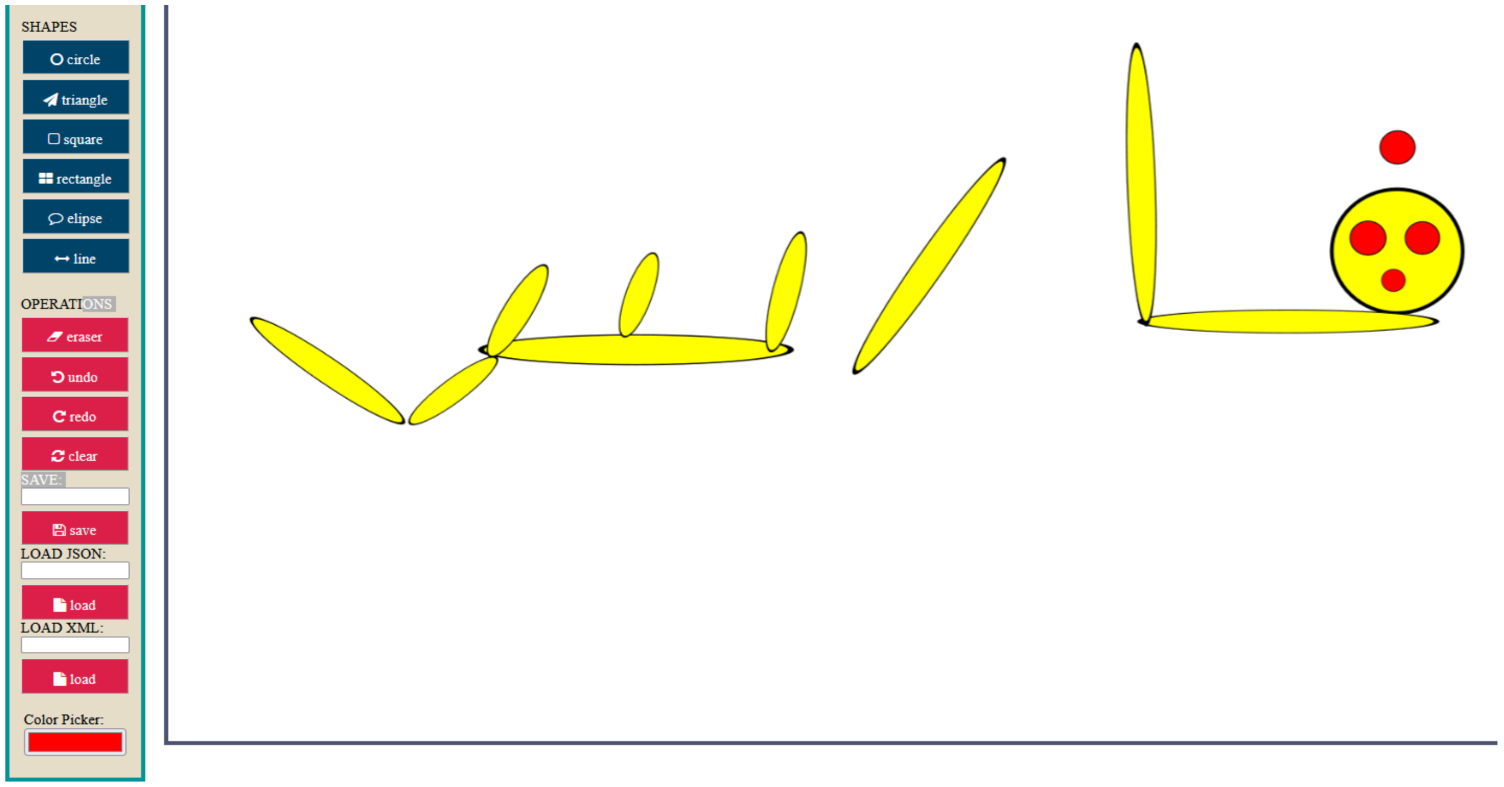
In front and back we convert from , to obj and json to transport Data Used that in Undo , Redo , Save ,Load.

Assumptions:

- When user make refresh ,user Start new window without undo , redo to last pictures
- Undo and redo can be used from start of program until now
- We used Equilateral triangle to express a triangle
- All shapes maintain their shape and the proportions of their sides relative to each other in case of resizing
- Clear button Start a new program same as refresh .

User Guide:

- Should Run The code of Back end using InJ
- Should Run angular (HTML , CSS,TYPESCRIPT)
- Open local Host 4200
- Start to Draw by click on shape and the shape automaticly.
  Drawn in center of the page
- Used any operations when he want to undo ,redo , clear, Erase



Vidoe Will BE with Pdf to show all Cons: