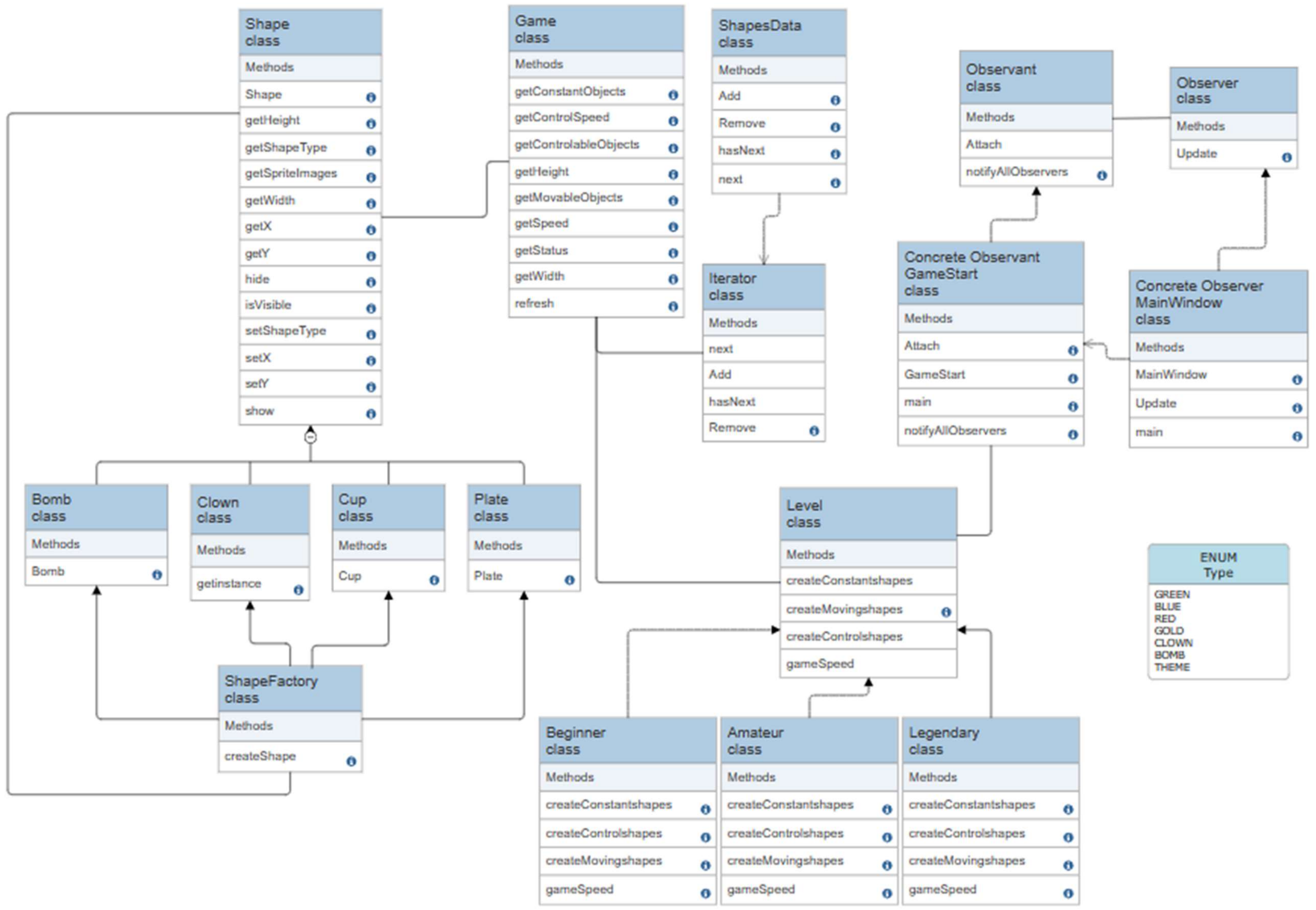
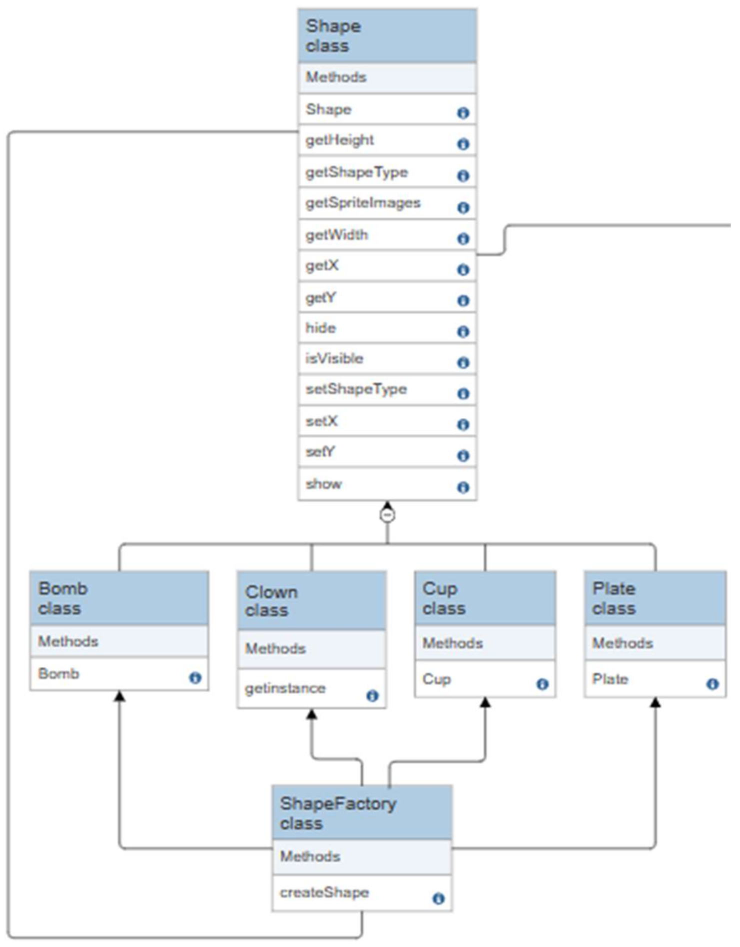
Final Lab Report

Students:
Youssef Mohamed Attia 8164
Karim Mohamed Almahrouky 8237
Abdelrahman Shaaban Saad 8041

```
if (intersectRight(clown, m)) {
                rightstackheight += m.getHeight();
                m.setX(clown.getX() + 165);
                m.setY(clown.getY() + 10 - rightstackheight);
                m.setycontrol(1);
                control.Add(m);
                moving.Remove(m);
                rightstack.Add(m);
                int x = rightstack.List().size();
                if (x >= 3) {
                   Type shapeType1 = rightstack.get(x - 1).getShapeType();
                   Type shapeType2 = rightstack.get(x - 2).getShapeType();
                   Type shapeType3 = rightstack.get(x - 3).getShapeType();
                   if (shapeType1 == shapeType2 && shapeType1 == shapeType3) {
                      for (int i = 1; i <= 3; i++) {
                         Shape temp = rightstack.get(x - i);
                         control.Remove((Shape) temp);
                         temp.setX((int) (Math.random() * width));
                         rightstackheight -= temp.getHeight();
                         moving.Add((Shape) temp);
                         temp.setycontrol(0);
                         temp.setY(0);
                         rightstack.Remove((Shape) temp);
                      }
                      score += 10;
                   }
                }
```


Shape class
Methods
Shape
getHeight
getShapeType
getSpriteImages
getWidth
getX
getY
hide
isVisible
setShapeType
setX
setY
show
Game class
Methods
getConstantObjects
getControlSpeed
getControlableObjects
getHeight
getMovableObjects
getSpeed
getStatus
getWidth
refresh
ShapesData class
Methods
Add
Remove
hasNext
next
Iterator class
Methods
next
Add
hasNext
Remove
Observant class
Methods
Attach
notifyAllObservers
Observer class
Methods
Update
Concrete Observant GameStart class
Methods
Attach
GameStart
main
notifyAllObservers
Concrete Observer MainWindow class
Methods
MainWindow
Update
main
Bomb class
Methods
Bomb
Clown class
Methods
getinstance
Cup class
Methods
Cup
Plate class
Methods
Plate
ShapeFactory class
Methods
createShape
Level class
Methods
createConstantshapes
createMovingshapes
createControlshapes
gameSpeed
ENUM Type
GREEN
BLUE
RED
GOLD
CLOWN
BOMB
THEME
Beginner class
Methods
createConstantshapes
createControlshapes
createMovingshapes
gameSpeed
Amateur class
Methods
createConstantshapes
createControlshapes
createMovingshapes
gameSpeed
Legendary class
Methods
createConstantshapes
createControlshapes
createMovingshapes
gameSpeed

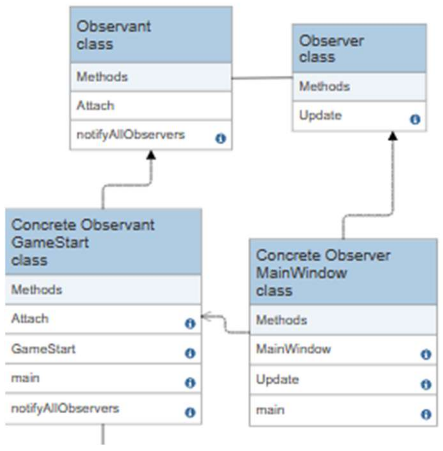
Factory & Singelton Design Pattern:



We used the factory design pattern to provide a decision-making class that returns one of several possible concrete subclasses (Bomb ,Clown ,Plate ,Cup) of an abstract shape class depending on the input provided.

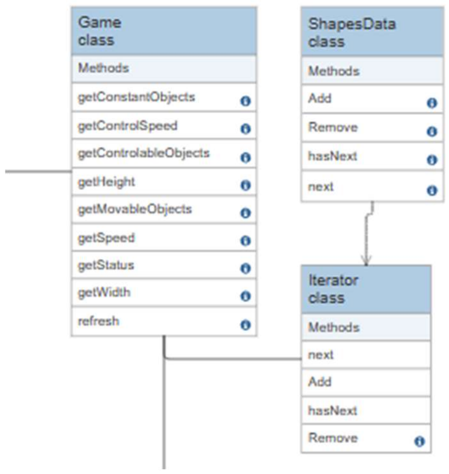
Singelton design pattern was used to ensure that only one instance of the shape Clown was created by making the class constructor private and calling it after checking if an instance is created yet.

Observer Design Pattern:



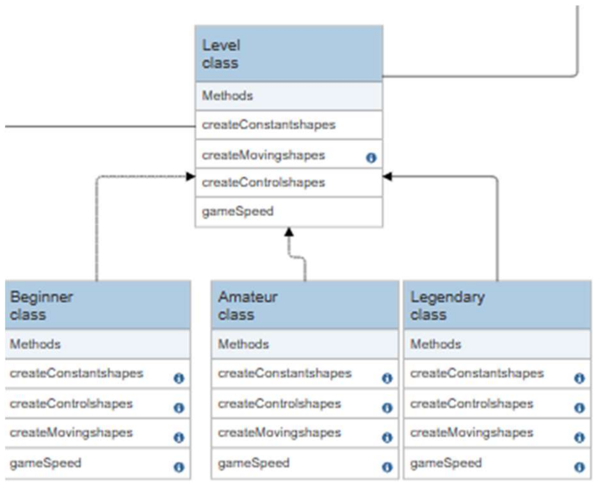
Our observer design pattern consists of 2 interfaces(observer, observant) , a concrete observer ”Main window” and a concrete observant ”Gamestart” so when the Gamestart state changes the observer class main window is notified and automatically updated.

Iterator Design Pattern:



Our Iterator design pattern consists of 1 interface class “Iterator” , a concrete Iterator ”ShapesData” and a client ”Game” so we used the iterator pattern to loop over all shapes collected in either clown hands or to edit those collected shapes while keeping the class private, so no changes are needed in client code.

Strategy Design Pattern:



We used the Strategy design pattern to provide different algorithms for a specific task (game level) and to facilitate adding another level at any time without interfering with the main game code in the Client class (Game).