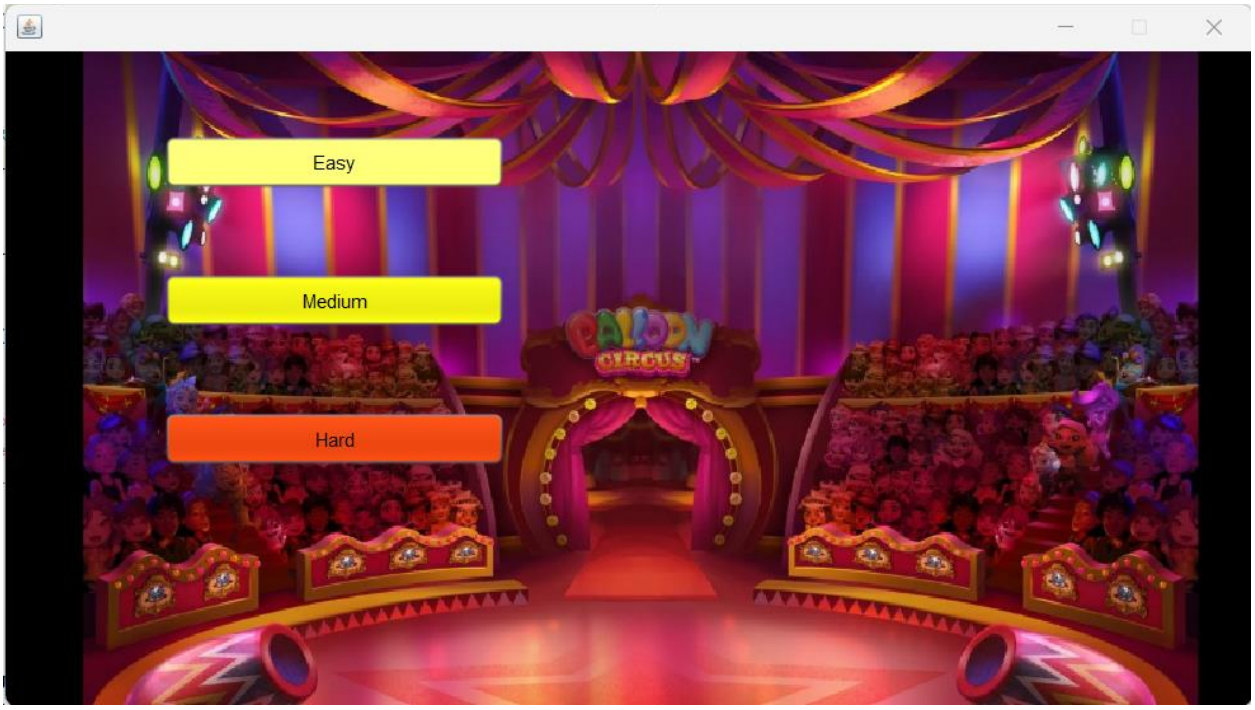
Ahmed Hendawy	8341
Ammar Elsamman	8023
Ahmed Elkhereby	8062
Mohamed Hisham	8079

Circus Of Balls Game

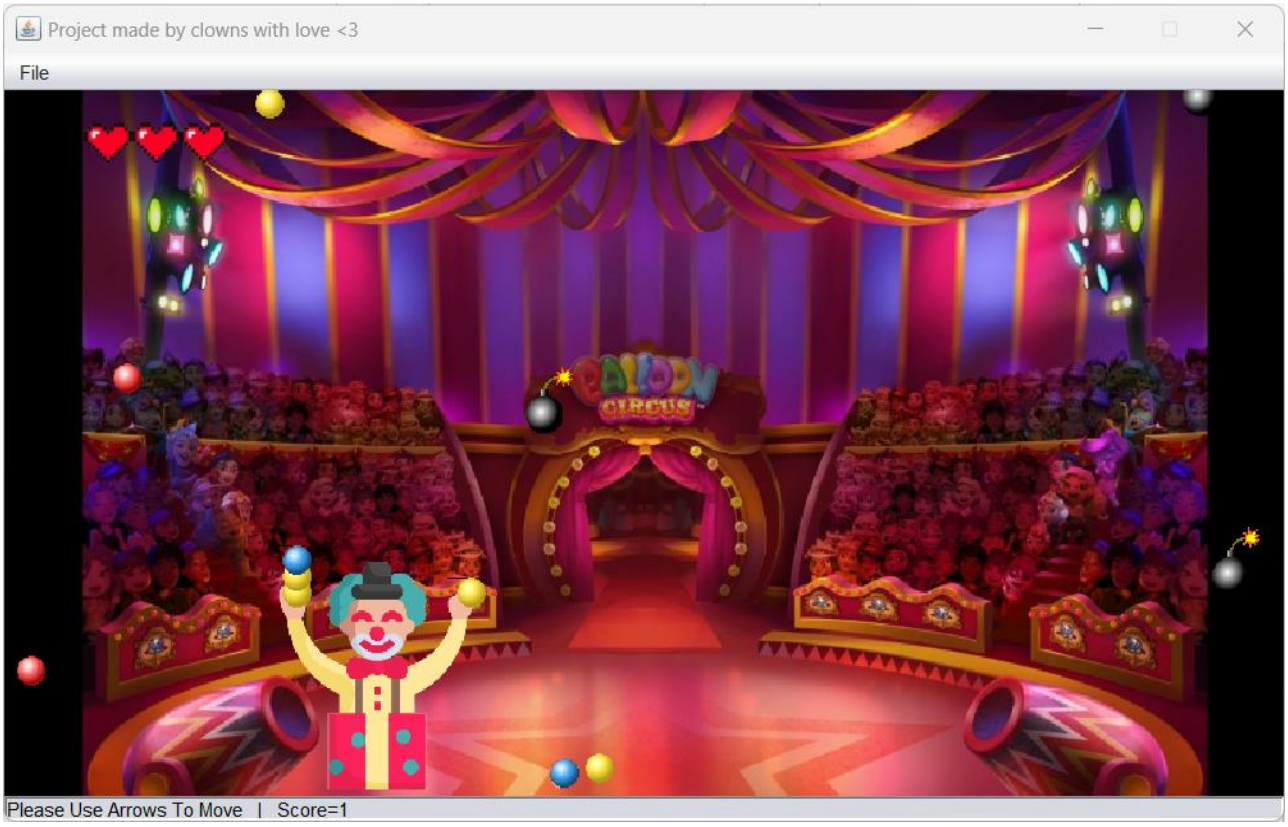
1) Design and how to play.

Single-player game where the clown collects two sets of balls with different colors that falls randomly, If he manages to collect three consecutive balls with same color, they vanish, and his score increases.

Game start menu



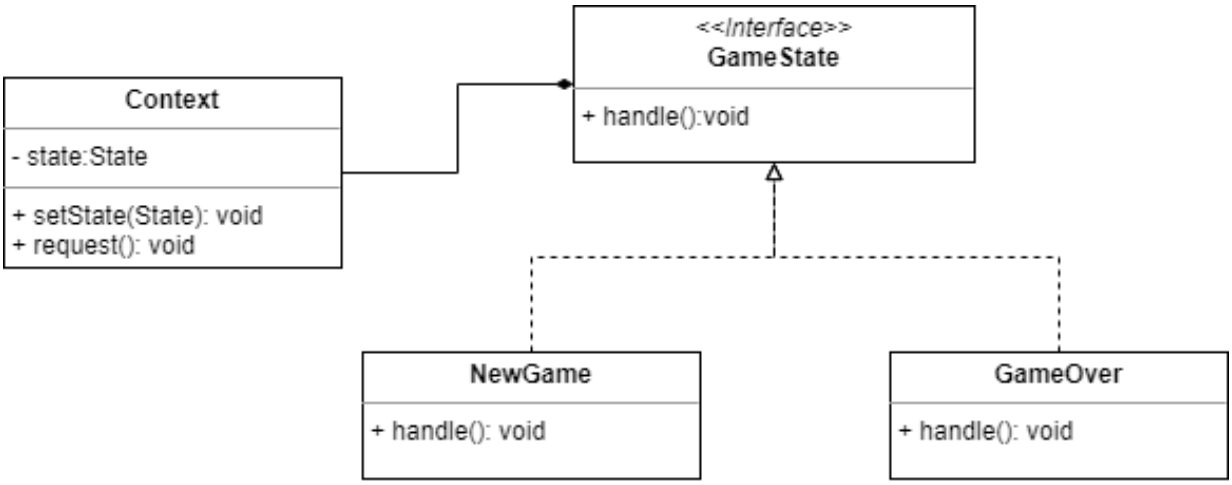
The difficulties change the speed of the balls falling.

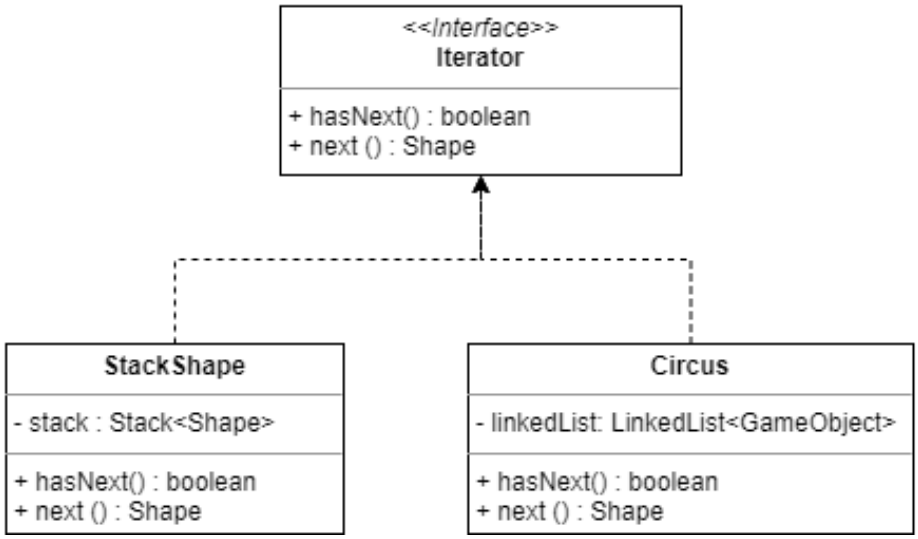


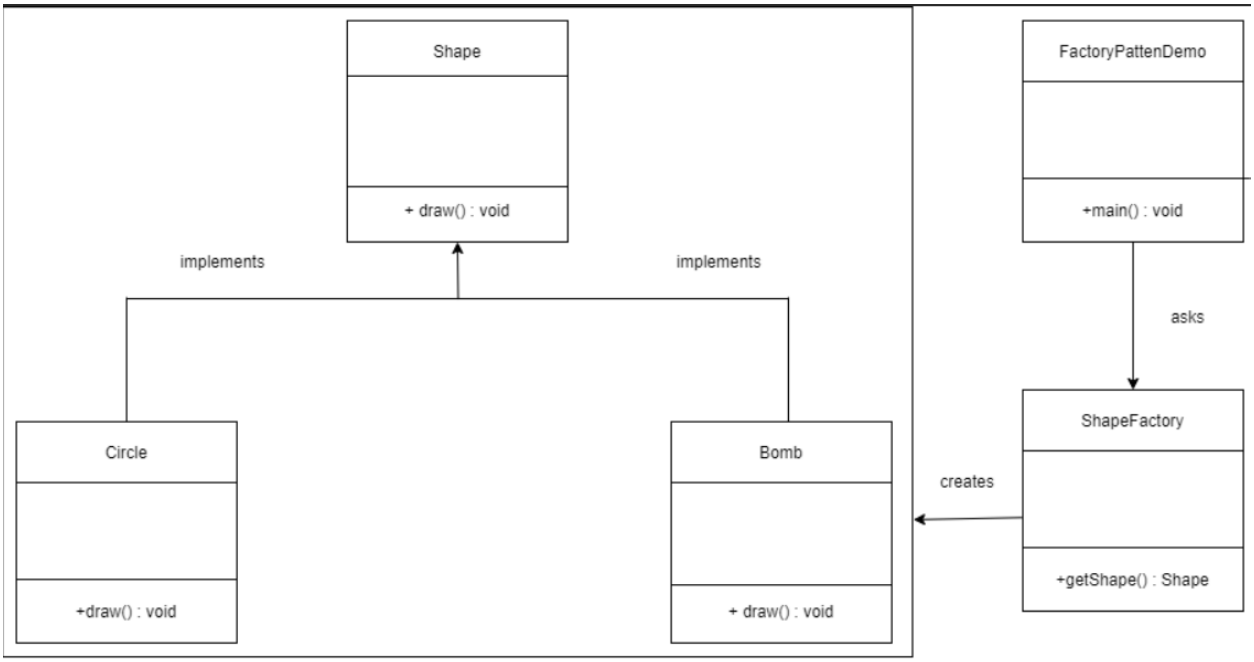
Score and lives

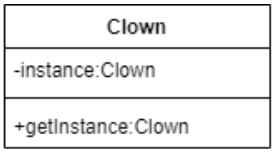
Once you collect 3 balls you get one point, and you start with 3 lives you lose lives when you receive a bomb. The game ends when you lose all your lives.

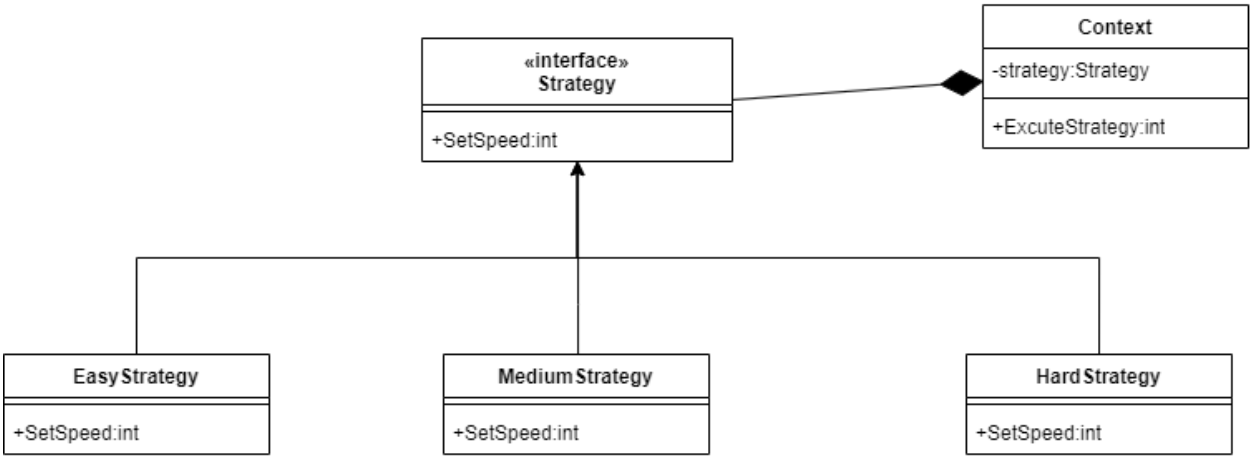
2) Class diagram






Shape
+ draw() : void
implements
implements
Circle
+draw() : void
Bomb
+ draw() : void
FactoryPattenDemo
+main() : void
asks
ShapeFactory
+getShape() : Shape
creates


Clown
-instance:Clown
+getInstance:Clown


«interface»
Strategy
+SetSpeed:int
Context
-strategy:Strategy
+ExcuteStrategy:int
Easy Strategy
+SetSpeed:int
Medium Strategy
+SetSpeed:int
Hard Strategy
+SetSpeed:int

1)Factory

Create the image of bomb, ball, and hearts.

```
public class ShapeFactory {

    Random rand = new Random();

    public ShapeFactory() {
    }

    public ImageObject createObject(String objectName) {
        if (objectName.equalsIgnoreCase("ball")) {
            String[] colors = {"Red", "Blue", "Yellow"};
            String path = colors[rand.nextInt(colors.length)] + "_Ball.png";
            return new Ball(0, 0, path, 1);
        } else if (objectName.equalsIgnoreCase("bomb")) {
            return new Bomb(0, 0, "Bomb.png", 0);
        } else if (objectName.equalsIgnoreCase("Heart")) {
            return new Heart(0, 0, "Heart.png");
        }
        return null;
    }
```

2)Strategy

Used to determine the speed of the game.

```
 */
public interface  Strategy {
    public int getSpeed();
}
```

3)State

Used to determine the state of the game.

```
public abstract class GameState {
    private boolean state;
    private Circus circus;

    public GameState(Circus c){
        this.circus =c;
    }

    public void setState(boolean state){
        this.state = state;
    }

    public Circus getCircus(){
        return this.circus;
    }

    public boolean getState(){
        return this.state;
    }

    public abstract void gameAction();
}
```

4)Iterator

handles the stack and linked lists.

```
public interface Iterator {

    void addObject(Object ball);

    Object removeObject(Object ball);

    boolean isEmpty();

    Object getPeekValue();

    int getCounter();

    void setCounter(int counter);

    int getSize();

    GameObject getValue(int index);

    LinkedList<GameObject> getIteratorTool();
}
```

5)Singleton

Create one instance of object of clown.