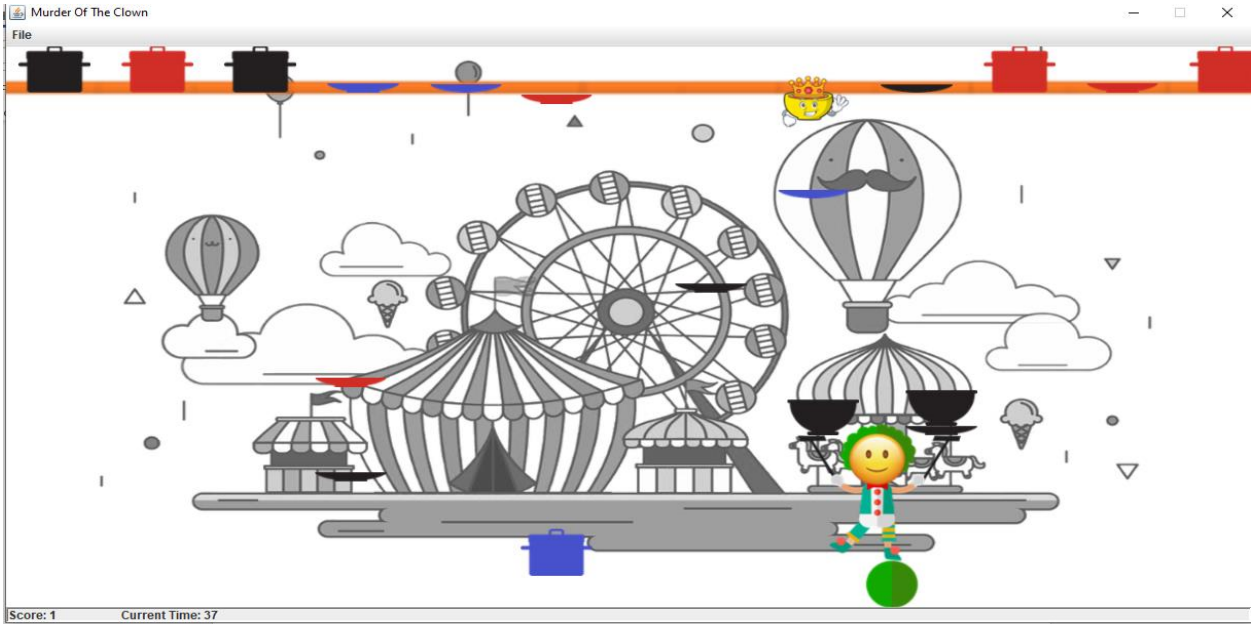
Design:

First of all, The game starts with a cool gui that is so friendly for user for all ages that can play music, choose the clown you want to play with, and of course play the actual game.

- Secondly, The game is about a clown who is trying so hard to catch three consecutive plates with the same colors but he can’t so he need your help to actually achieve his lifetime dream and he will be grateful all his life (till refresh return false)



- The goal is to achieve a score of 10 before the time runs out which is 2 minutes, there is also some special plate that multiply your score for a small period of time to help you defeat your greatest enemies and fears at the same time.

- Lastly the user can watch his last played game just by clicking on the Replay Button.

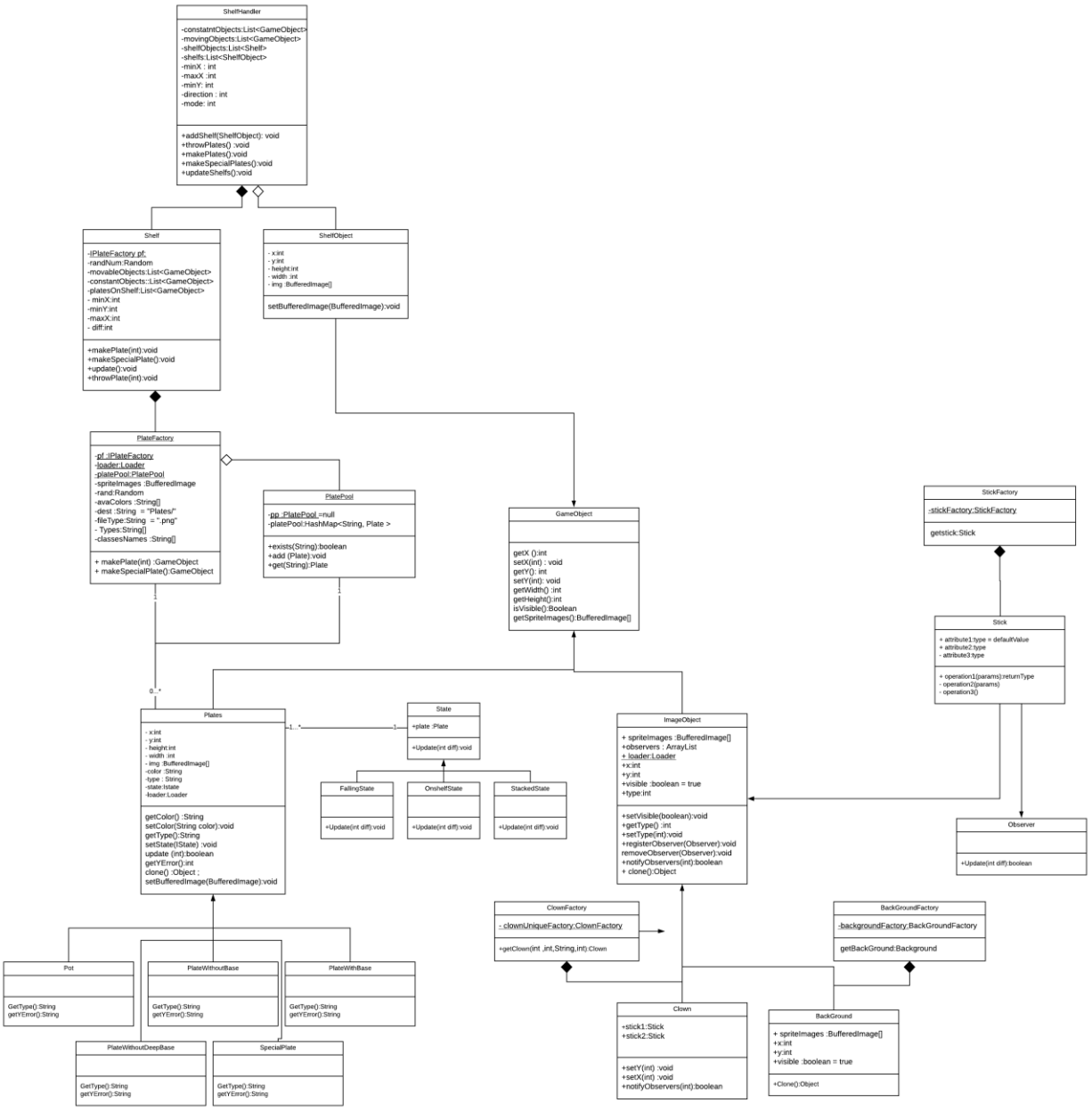
- **The used design Patterns** are:

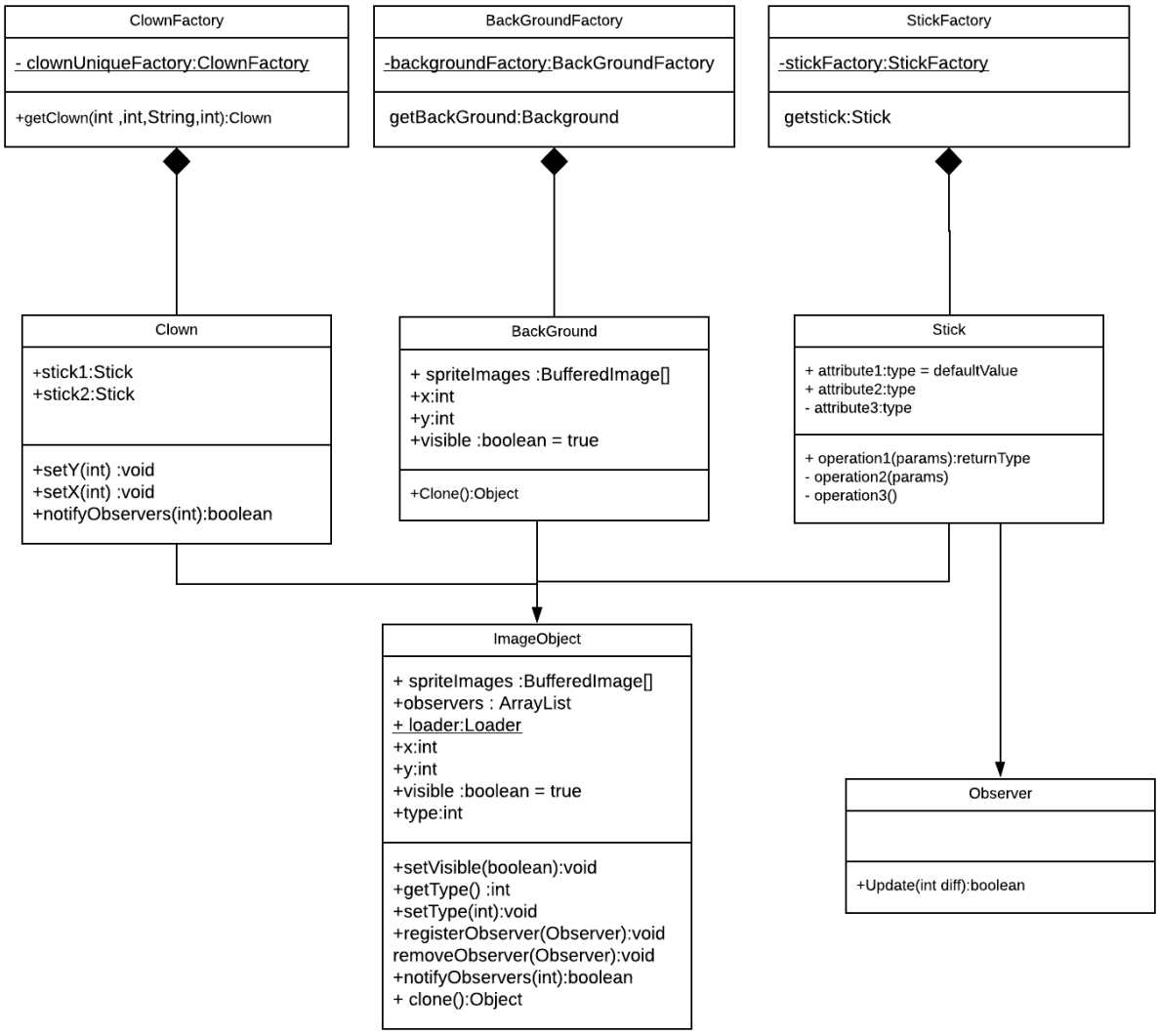
1. Singleton 2. Factory and Pool 3. Iterator

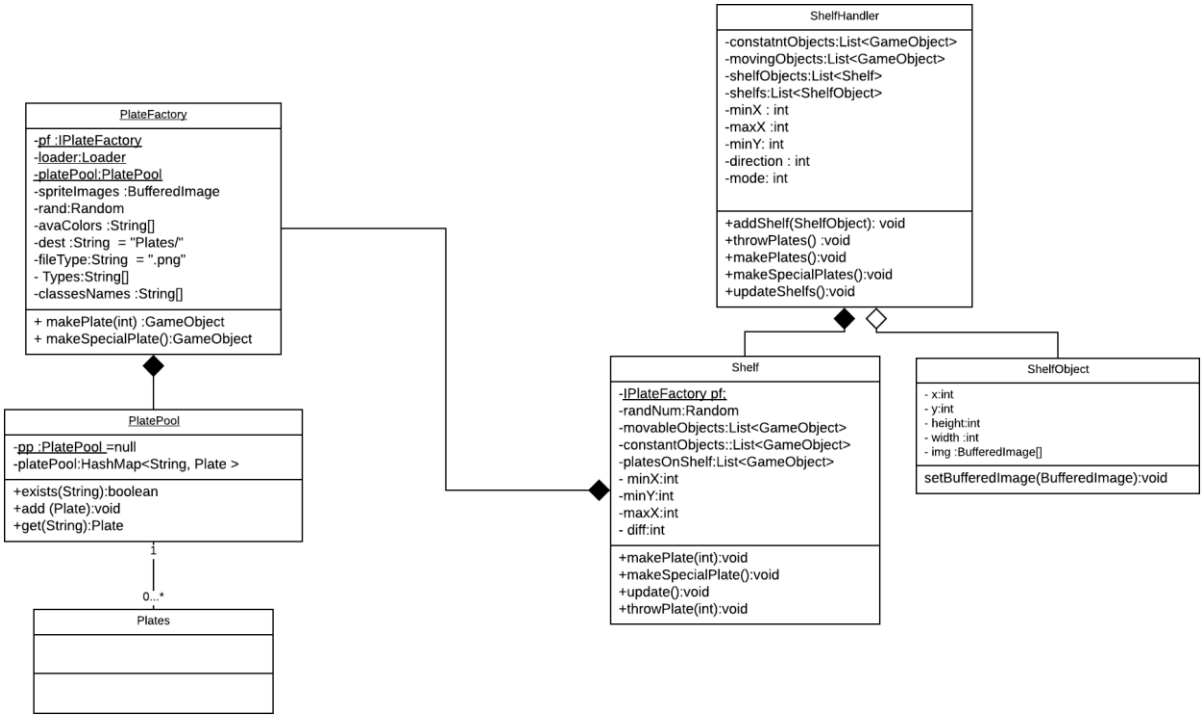
4. Dynamic Linkage 5. Snapshot 6. State 7. Strategy

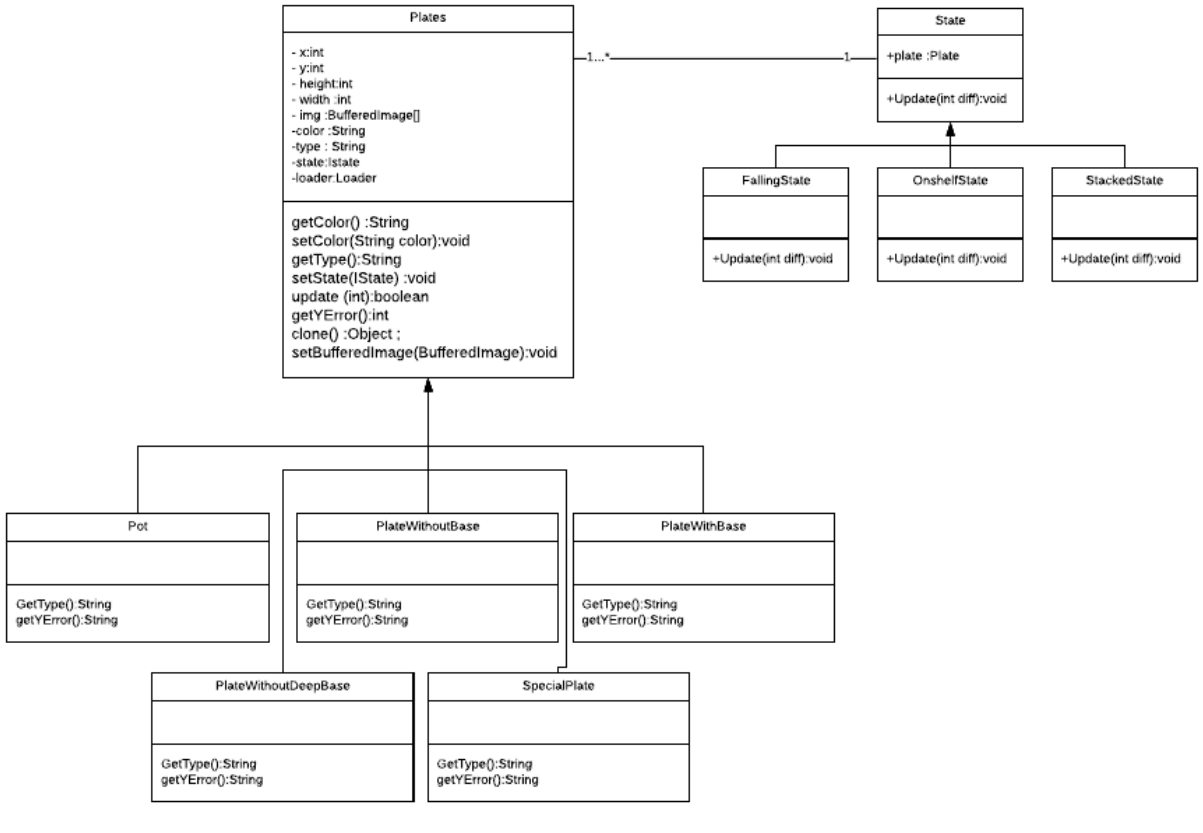
8. Flyweight 9. Observer 10. Façade 11.Command

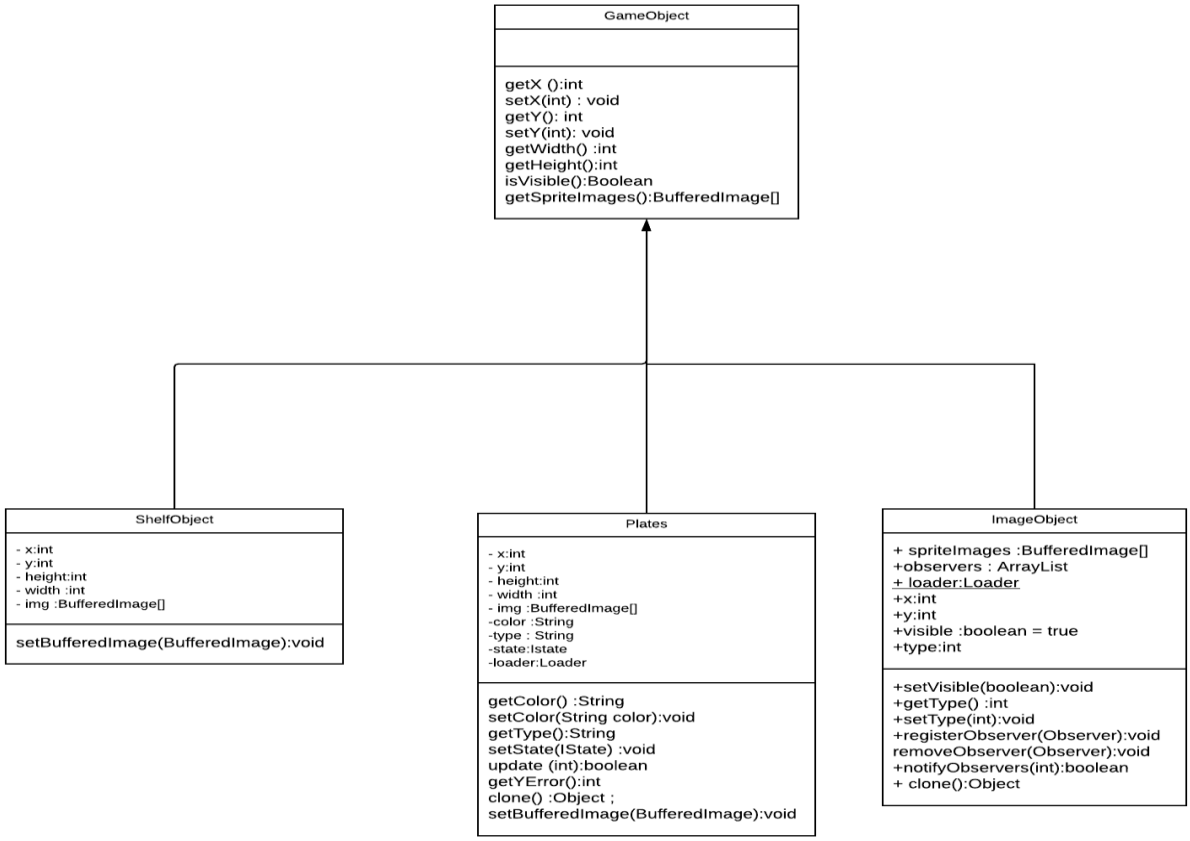
Class Diagram:



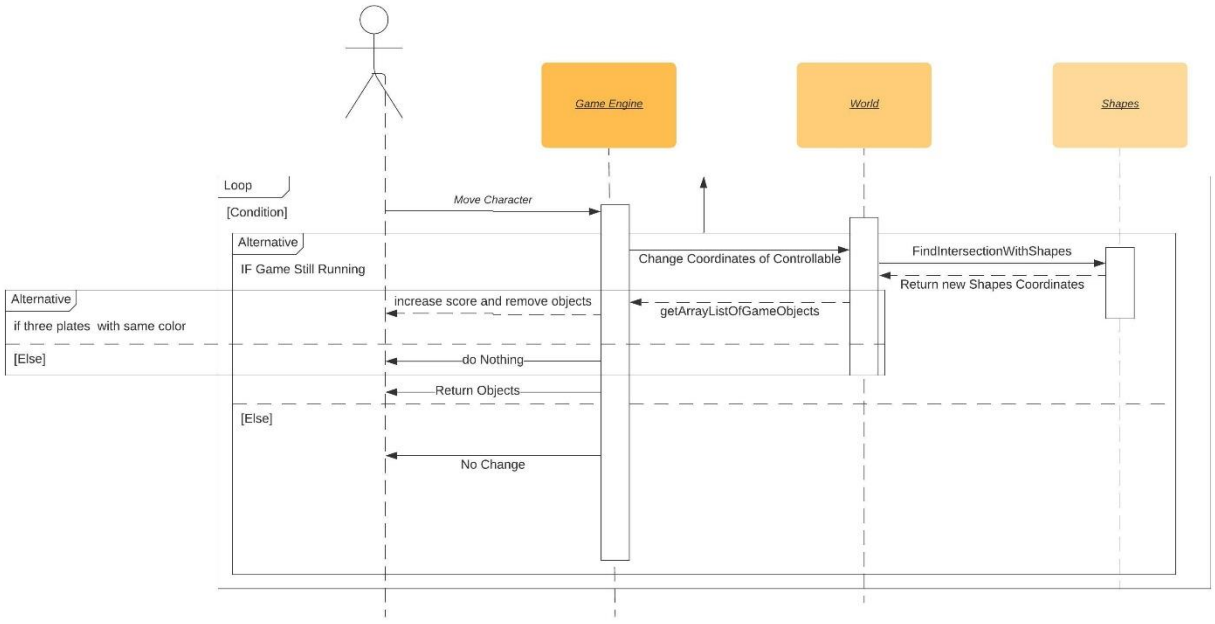

ClownFactory
- clownUniqueFactory:ClownFactory
+getClown(int ,int,String,int):Clown
BackGroundFactory
-backgroundFactory:BackGroundFactory
getBackGround:Background
StickFactory
-stickFactory:StickFactory
getstick:Stick
Clown
+stick1:Stick
+stick2:Stick
+setY(int) :void
+setX(int) :void
+notifyObservers(int):boolean
BackGround
+ spriteImages :BufferedImage[]
+x:int
+y:int
+visible :boolean = true
+Clone():Object
Stick
+ attribute1:type = defaultValue
+ attribute2:type
- attribute3:type
+ operation1(params):returnType
- operation2(params)
- operation3()
ImageObject
+ spriteImages :BufferedImage[]
+observers : ArrayList
+ loader:Loader
+x:int
+y:int
+visible :boolean = true
+type:int
+setVisible(boolean):void
+getType() :int
+setType(int):void
+registerObserver(Observer):void
removeObserver(Observer):void
+notifyObservers(int):boolean
+ clone():Object
Observer
+Update(int diff):boolean


ShelfHandler
-constatntObjects:List<GameObject>
-movingObjects:List<GameObject>
-shelfObjects:List<Shelf>
-shelfs:List<ShelfObject>
-minX : int
-maxX :int
-minY: int
-direction : int
-mode: int
+addShelf(ShelfObject): void
+throwPlates() :void
+makePlates():void
+makeSpecialPlates():void
+updateShelfs():void
PlateFactory
-pf :IPlateFactory
-loader:Loader
-platePool:PlatePool
-spriteImages :BufferedImage
-rand:Random
-avaColors :String[]
-dest :String = "Plates/"
-fileType:String = ".png"
- Types:String[]
-classesNames :String[]
+ makePlate(int) :GameObject
+ makeSpecialPlate():GameObject
PlatePool
-pp :PlatePool =null
-platePool:HashMap<String, Plate >
+exists(String):boolean
+add (Plate):void
+get(String):Plate
1
0...*
Plates
Shelf
-IPlateFactory pf:
-randNum:Random
-movableObjects:List<GameObject>
-constantObjects::List<GameObject>
-platesOnShelf:List<GameObject>
- minX:int
-minY:int
-maxX:int
- diff:int
+makePlate(int):void
+makeSpecialPlate():void
+update():void
+throwPlate(int):void
ShelfObject
- x:int
- y:int
- height:int
- width :int
- img :BufferedImage[]
setBufferedImage(BufferedImage):void


Plates
- x:int
- y:int
- height:int
- width :int
- img :BufferedImage[]
-color :String
-type : String
-state:IState
-loader:Loader
getColor() :String
setColor(String color):void
getType():String
setState(IState) :void
update (int):boolean
getYError():int
clone() :Object ;
setBufferedImage(BufferedImage):void
1...*
1
State
+plate :Plate
+Update(int diff):void
FallingState
+Update(int diff):void
OnshelfState
+Update(int diff):void
StackedState
+Update(int diff):void
Pot
GetType():String
getYError():String
PlateWithoutBase
GetType():String
getYError():String
PlateWithBase
GetType():String
getYError():String
PlateWithoutDeepBase
GetType():String
getYError():String
SpecialPlate
GetType():String
getYError():String



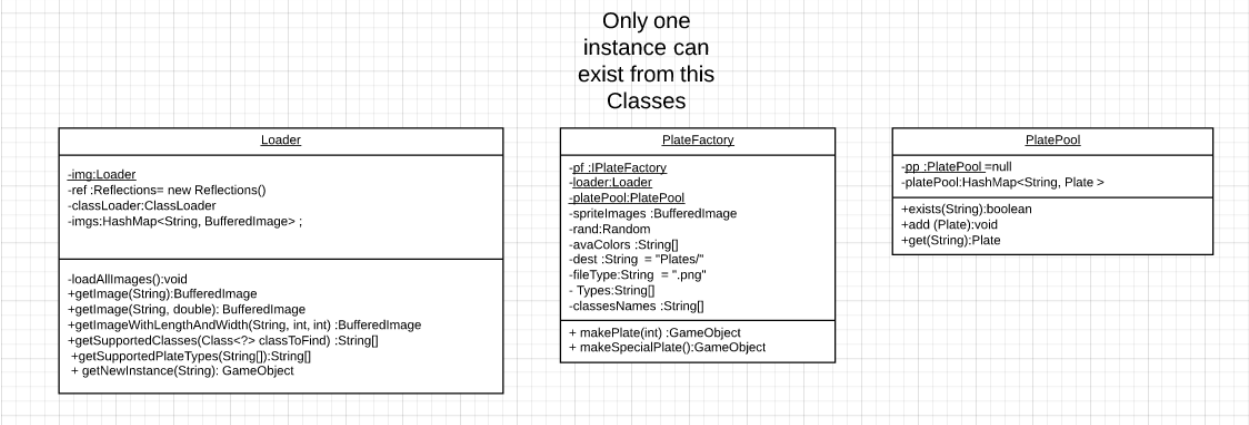
Sequence Diagram:



1) Singelton :

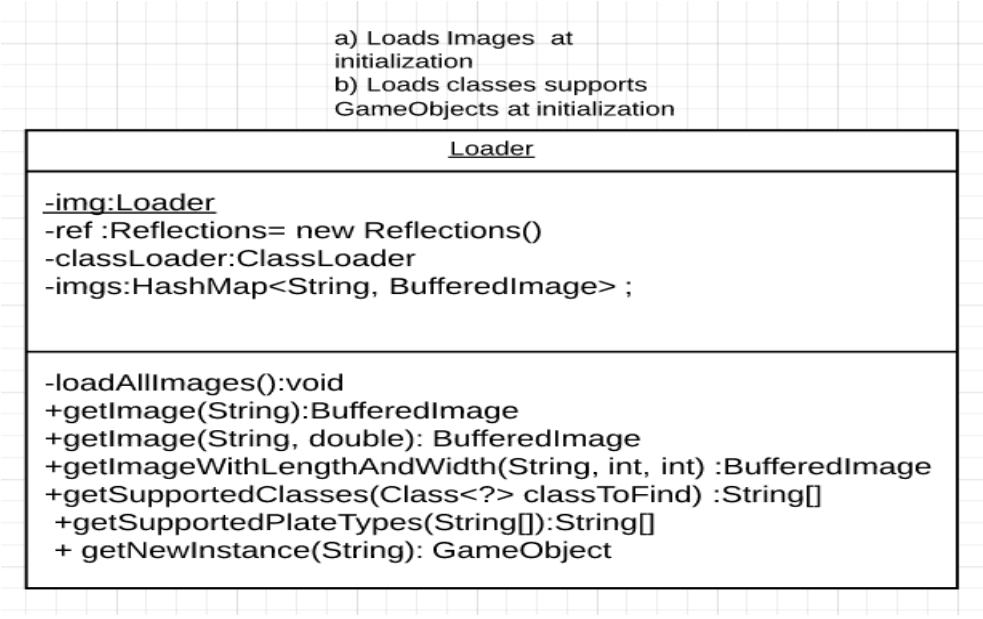
The following classes were written in Singelton pattern :

a) PlateFactory
b) Loader
c) PlatePool



2) Dynamic Linkage : (Loader.class)

a) Dynamically loads class names and Full Package Pathes that implements GameObject and extends Plate
b) Dynamically loads Images that exists in the resources directory defined in project structure



3) State : (States.class)

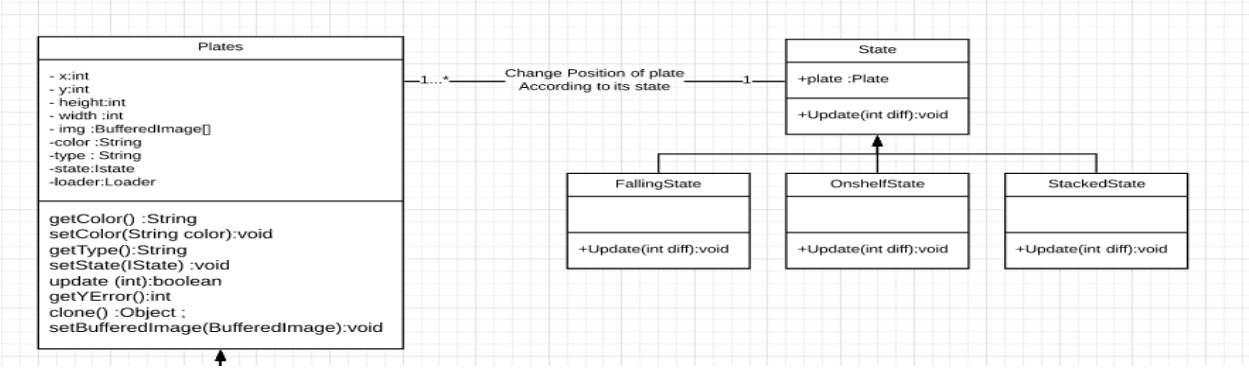
Each plate has a different state that defines its way of moving in screen .

States Available :

a) MovingOnShelf State

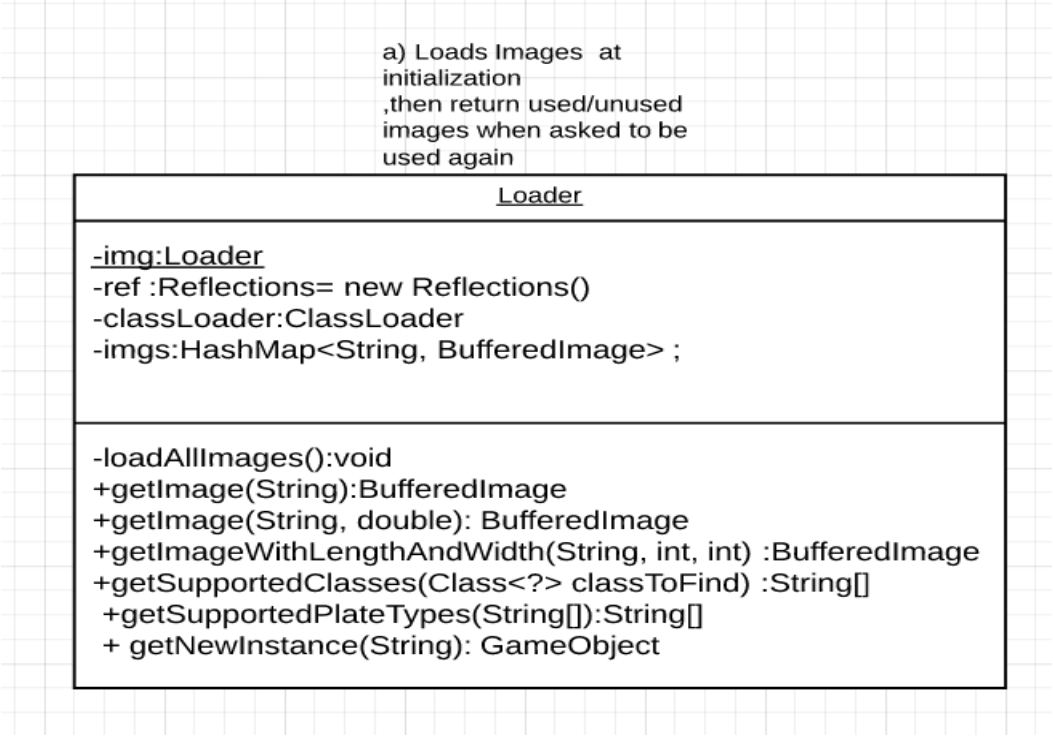
b) Falling State

c) Stacked (moving with the clown) State



4) Fly Weight : (Loader.class)

Each Time an image is needed is used from a preloaded HashMap where more than one plate would have the same BufferedImage Object

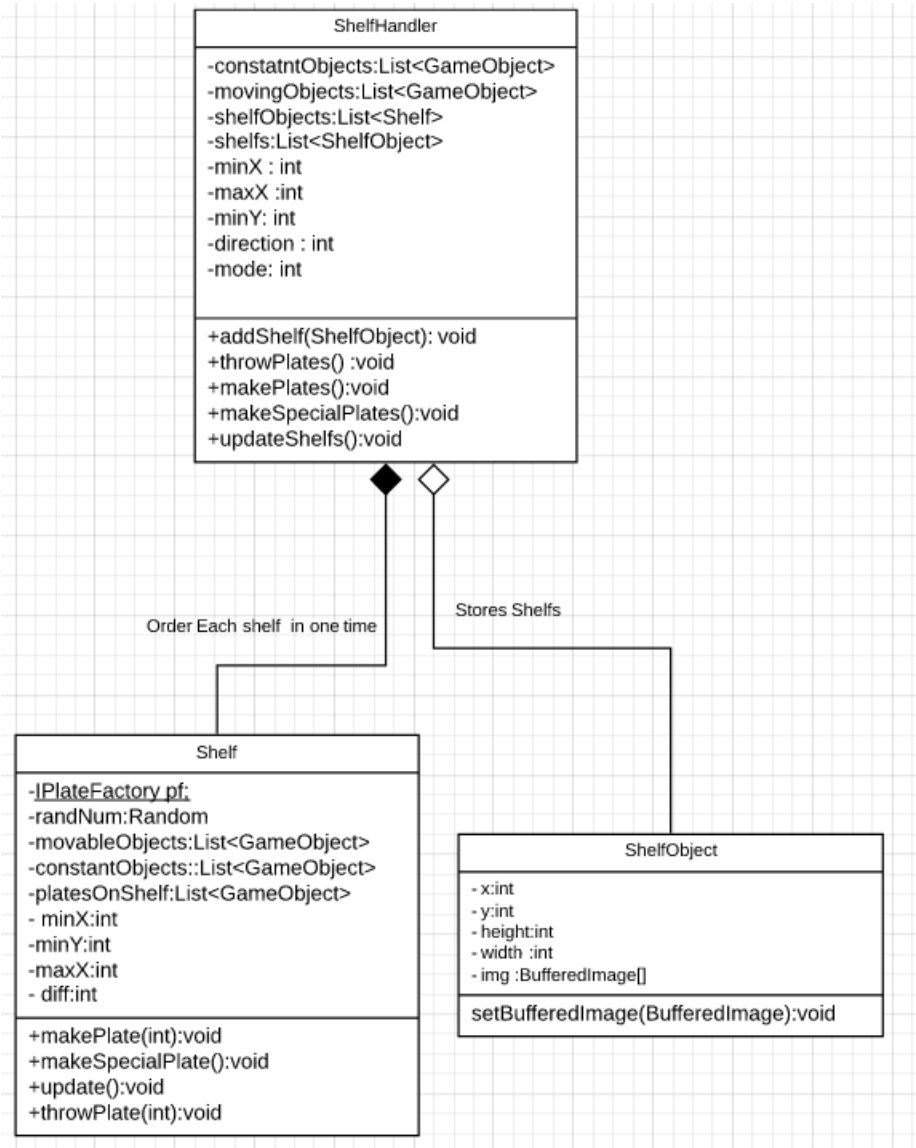


5) Command : (ShelfHandler.class->Shelf.class)

a) ShelfHandler class holds more than one Shelf

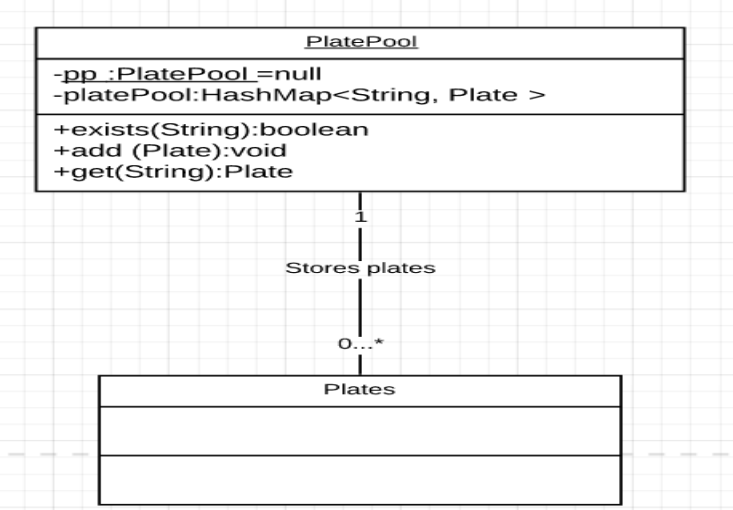
b) Once throw or make plate is ordered from Gui class , it orders all the shelfs it holds to make/throw new plates at the same time

c) each shelf orders the plate factory to make a new plate for him .



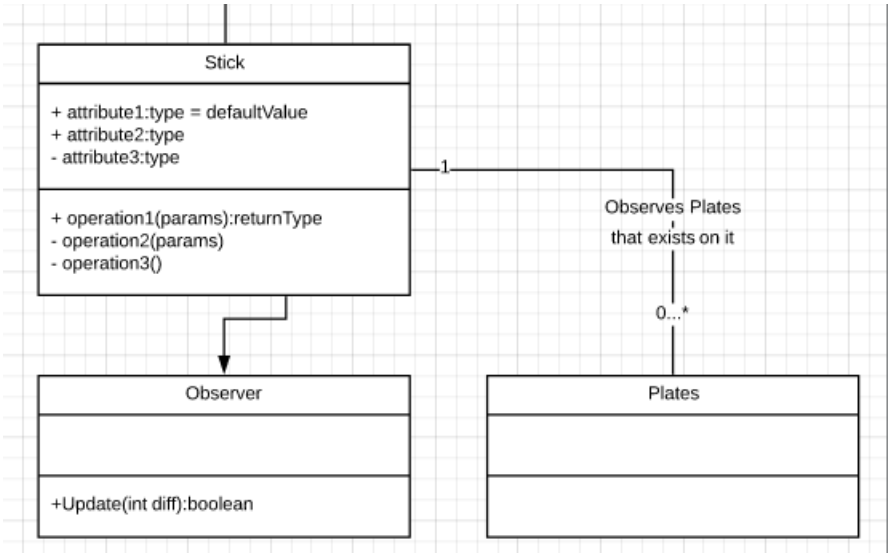
6) Pool : (PlatePool.class)

a) Carries unused Plate Objects
b) When a plate reaches maximum y in screen is added to the pool
c) Each time PlateFactory.makeplate method is called , the factory checks first if there's a free unused plate from the pool
d) If it exists it returns that Object with new x ,y , else it makes a new plate



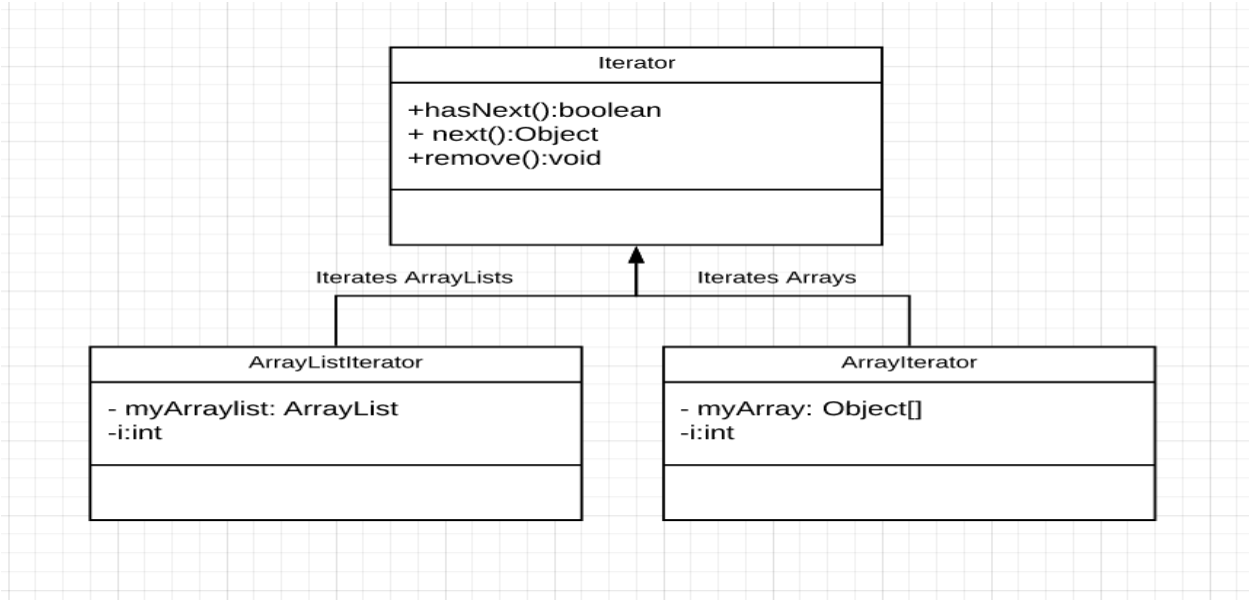
7) Observer:

a) Made Clown and Sticks Movements Trigger All Movement Of All Items Observing Them.



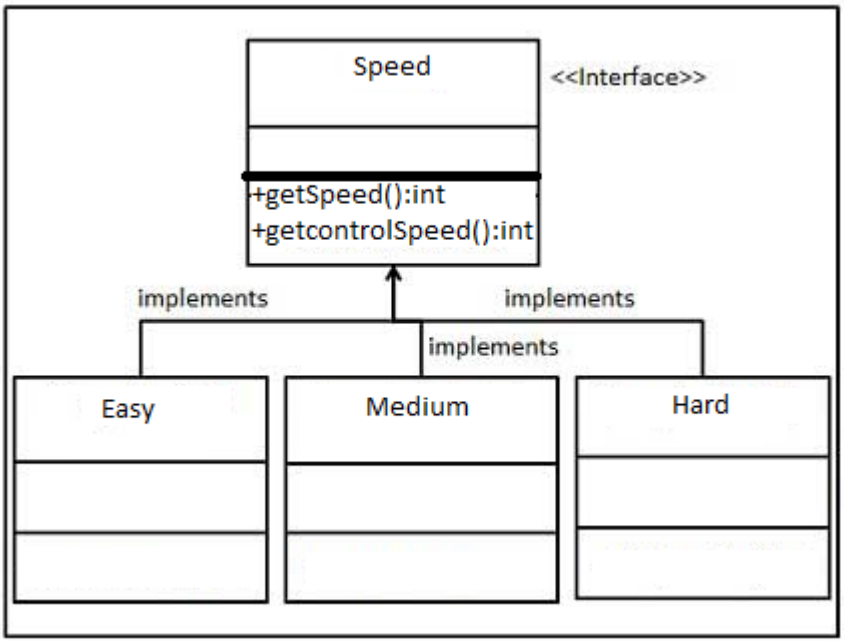
8) Iterator:

a) Used Iterator To Iterate Over ArrayLists and Arrays Alike like for Constant, Movable, And Controllable GameObjects ArrayList.



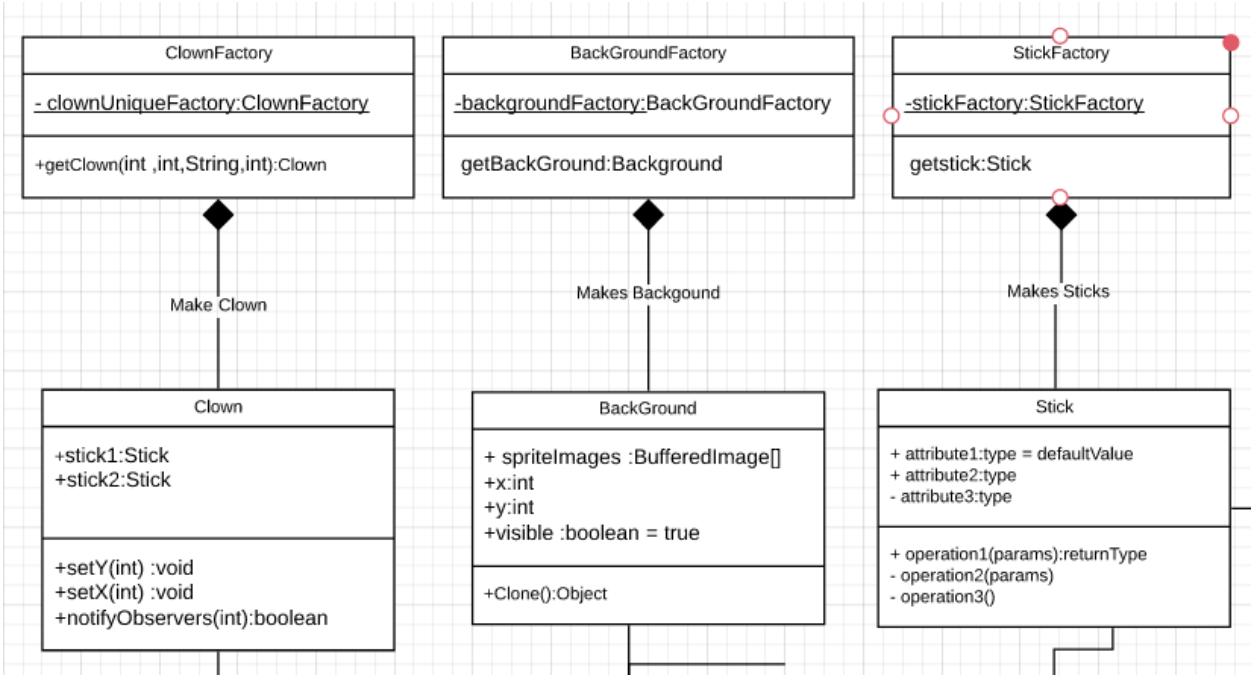
9) Strategy:

a) Controlled The Behavior of The Speed Of Each World and Made It More Flexible To Add new Worlds With Different Behaviors For The Speed For Each Of The Controllable Objects and GameObjects.



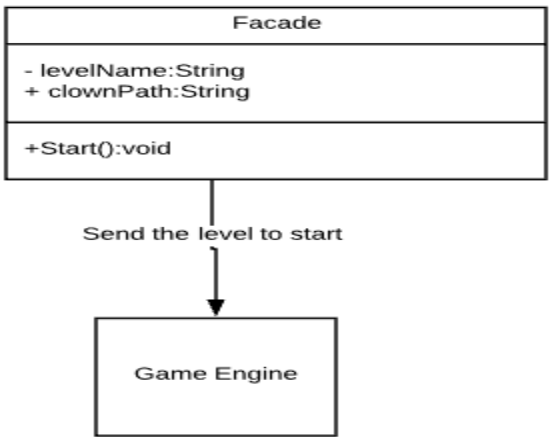
10)Factory:

a) Made A Factory To Produce Clowns, Sticks, Plates, BackGround Images.



11)Façade:

a) takes which level the user want to play it and start it



12) SnapShot:

a) Takes a copy for every object in the game every refresh and save it in an array of arraylists then be accessible in Replay so the **USER CAN WATCH HIS LATST PLAYED GAME.**