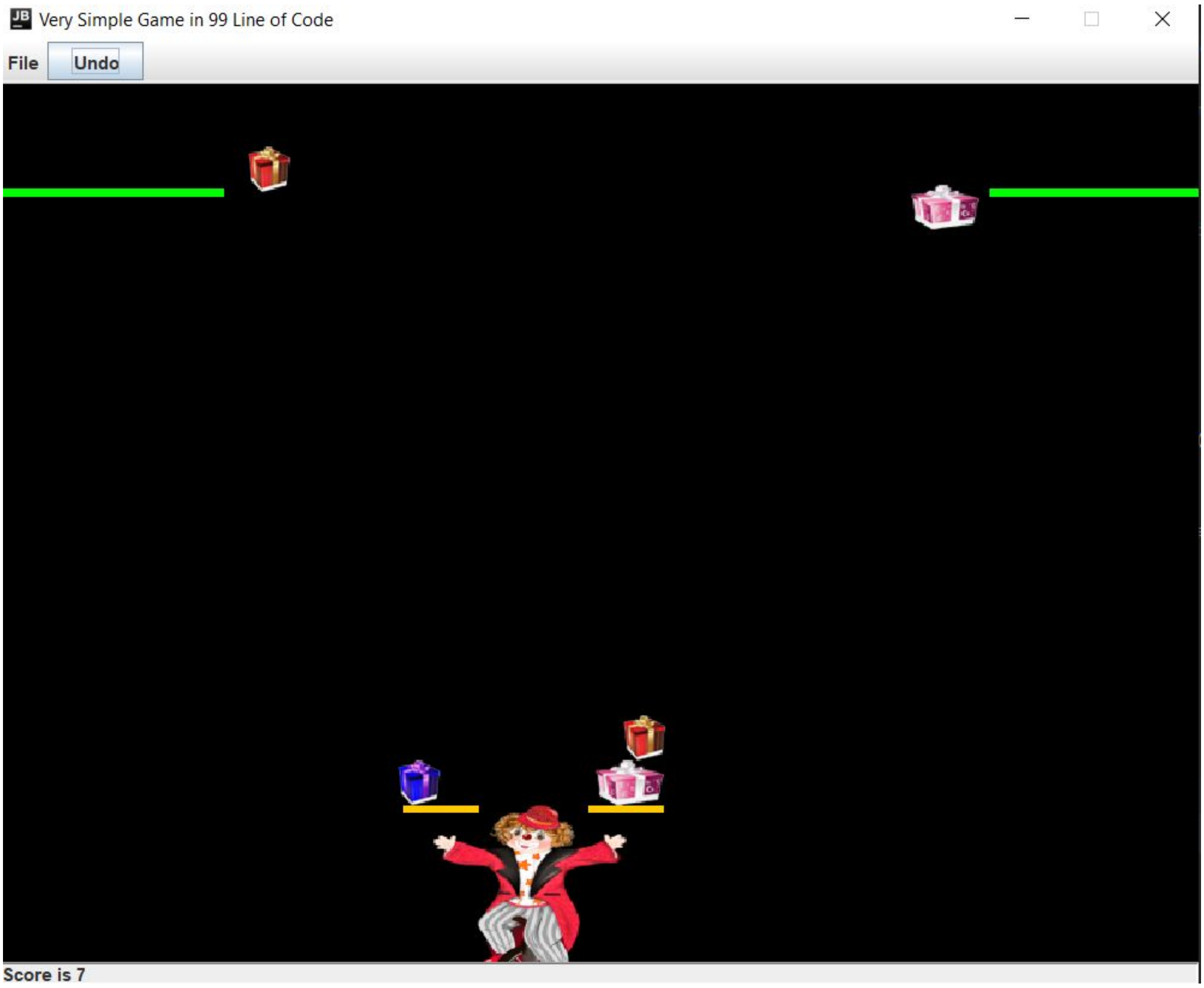
Circus of The Plates



Game Description :

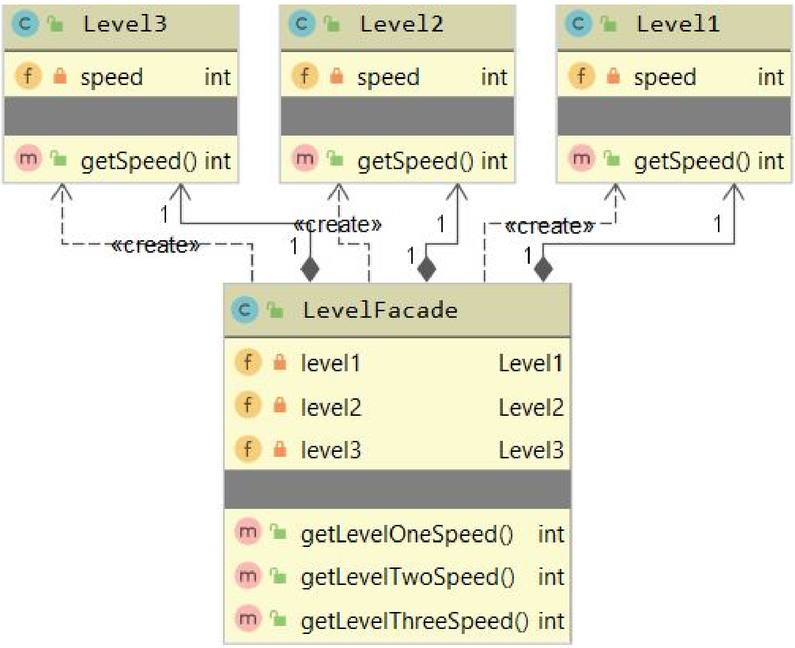
There are falling plates and player follow them to collect three consecutives plates similar in color (size is not required) when he collect them he score a point and he reached a particular number of points in a time equals to 30 seconds so he wins if not he loses

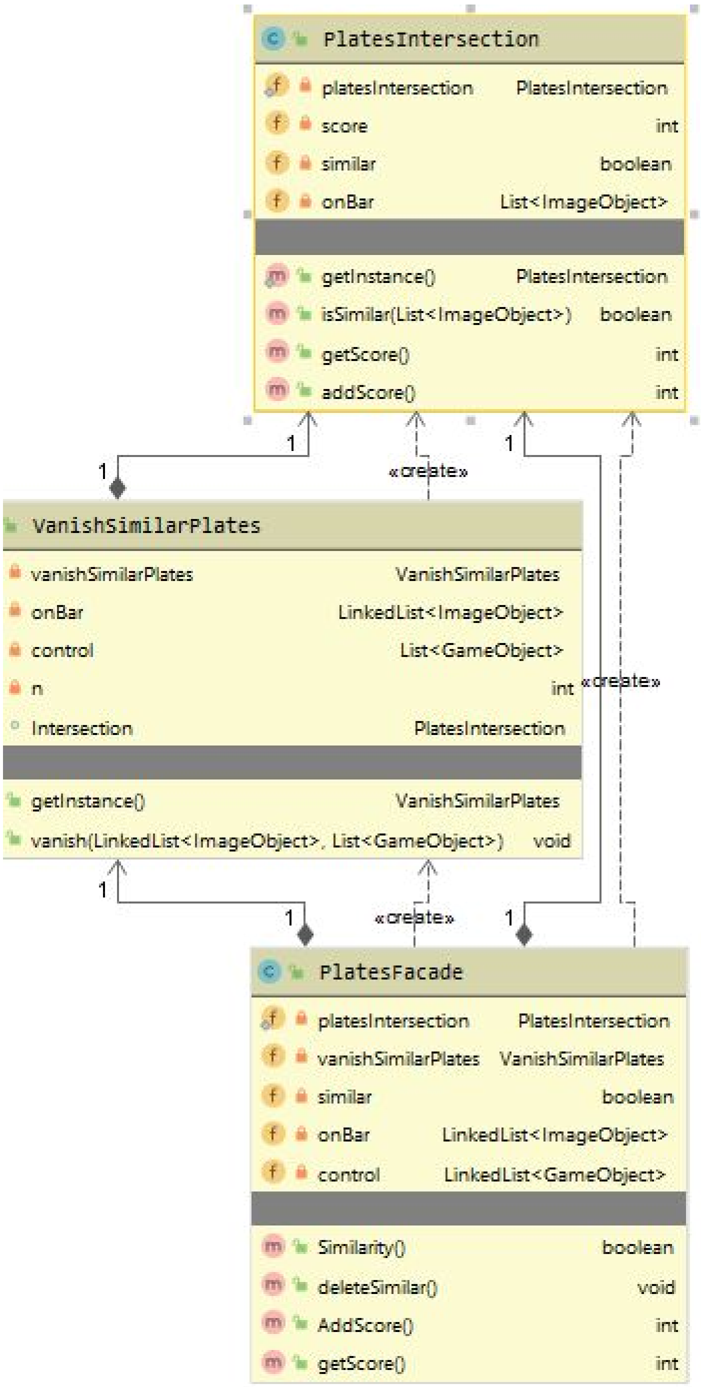
The game has three levels each level is different from the other in the speed of falling plates .

The game supports that the player can watch the whole game after he finishes (replay)

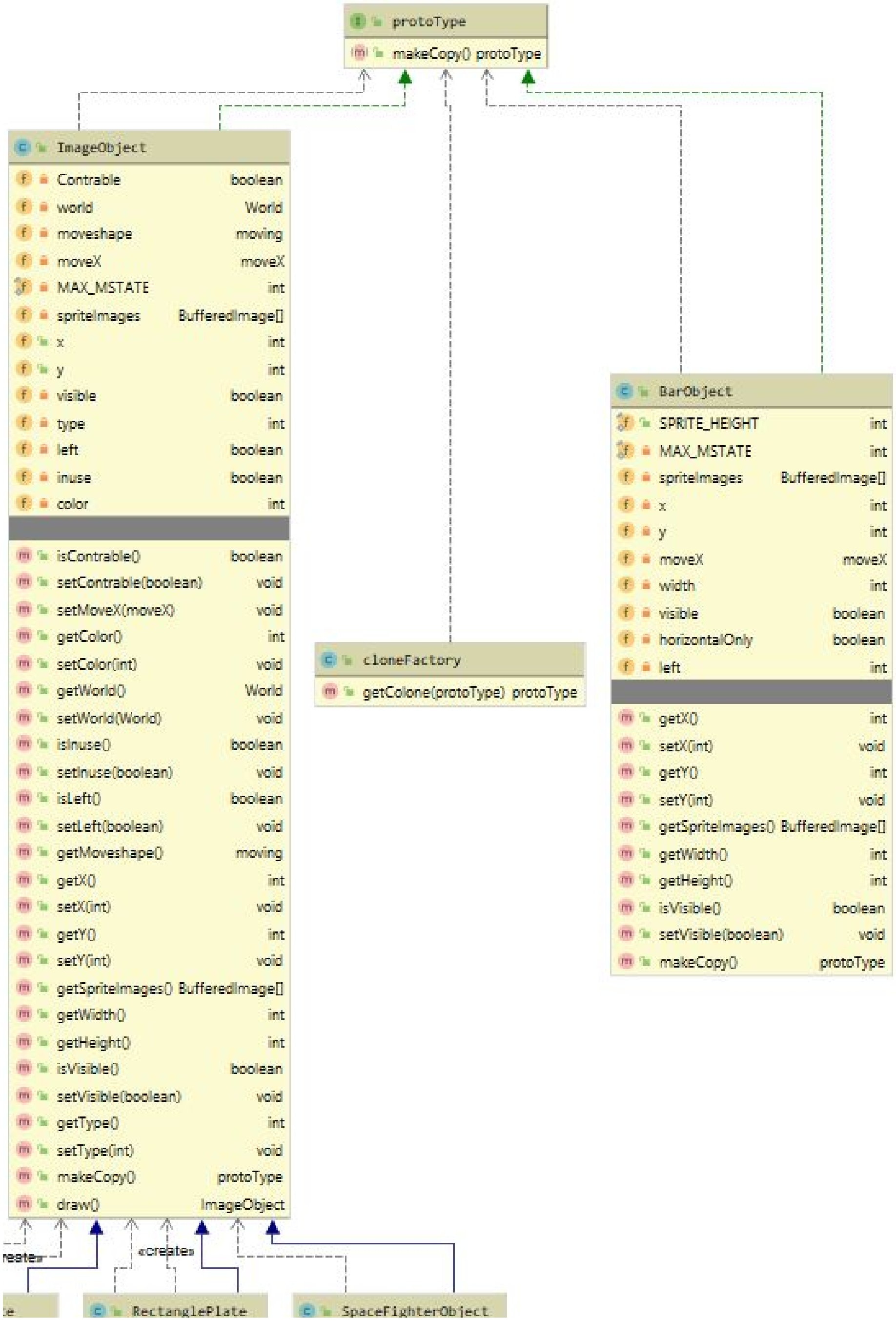
Uml diagram for Design Patterns in the project

1-facade design pattern
Used for two things
-> choosing the level
-> remove the similar plates and increase the score and showing score


Level3
speed int
getSpeed() int
Level2
speed int
getSpeed() int
Level1
speed int
getSpeed() int
«create»
«create»
«create»
1
1
1
1
1
1
LevelFacade
level1 Level1
level2 Level2
level3 Level3
getLevelOneSpeed() int
getLevelTwoSpeed() int
getLevelThreeSpeed() int

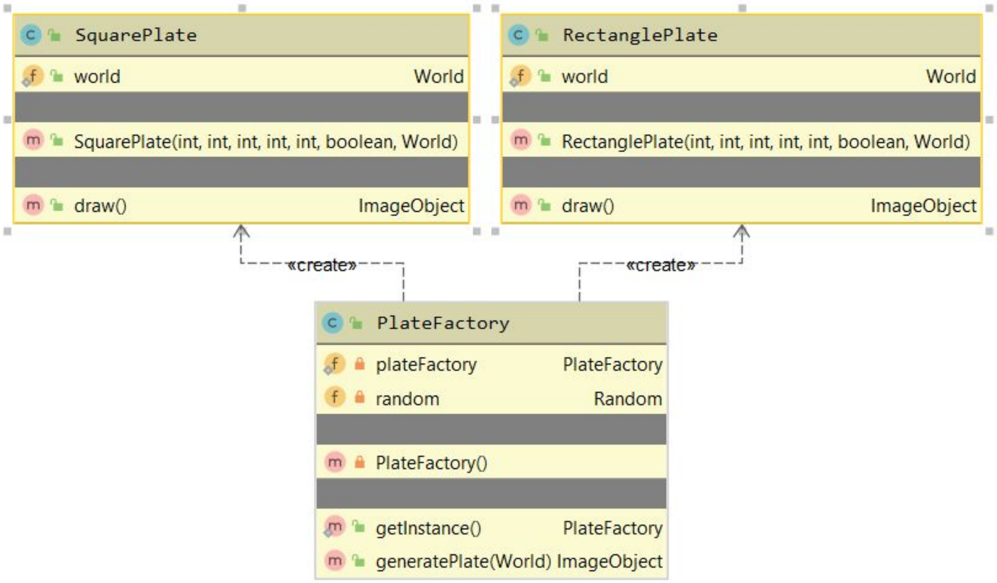

PlatesIntersection
platesIntersection PlatesIntersection
score int
similar boolean
onBar List<ImageObject>
getInstance() PlatesIntersection
isSimilar(List<ImageObject>) boolean
getScore() int
addScore() int
1
1
«create»
1
VanishSimilarPlates
vanishSimilarPlates VanishSimilarPlates
onBar LinkedList<ImageObject>
control List<GameObject>
n int
«create»
Intersection PlatesIntersection
getInstance() VanishSimilarPlates
vanish(LinkedList<ImageObject>, List<GameObject>) void
1
1
«create»
1
PlatesFacade
platesIntersection PlatesIntersection
vanishSimilarPlates VanishSimilarPlates
similar boolean
onBar LinkedList<ImageObject>
control LinkedList<GameObject>
Similarity() boolean
deleteSimilar() void
AddScore() int
getScore() int

2 - prototype design pattern
Instead of creating objects by word “new” and lose time for that we copy the objects to a new object in runtime
Uml for that



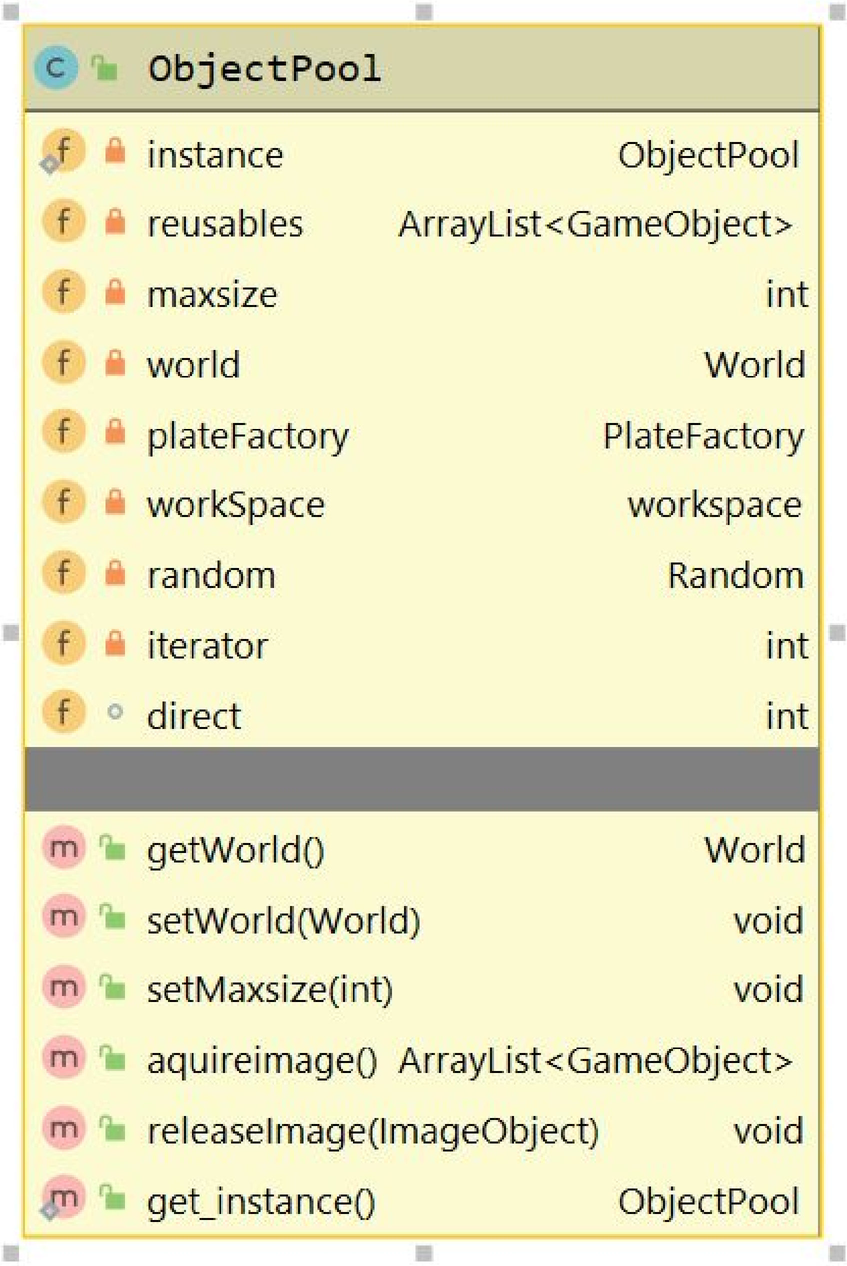
3- factory design pattern

Creating objects by random whether rectangle or square and for color also whether blue , pink or red


SquarePlate
world World
SquarePlate(int, int, int, int, int, boolean, World)
draw() ImageObject
RectanglePlate
world World
RectanglePlate(int, int, int, int, int, boolean, World)
draw() ImageObject
«create»
«create»
PlateFactory
plateFactory PlateFactory
random Random
PlateFactory()
getInstance() PlateFactory
generatePlate(World) ImageObject

4-object pool design pattern

Created for using only twenty plates only once the plates you see on the screen they return to the pool and we use it again “it replaces Flyweight “

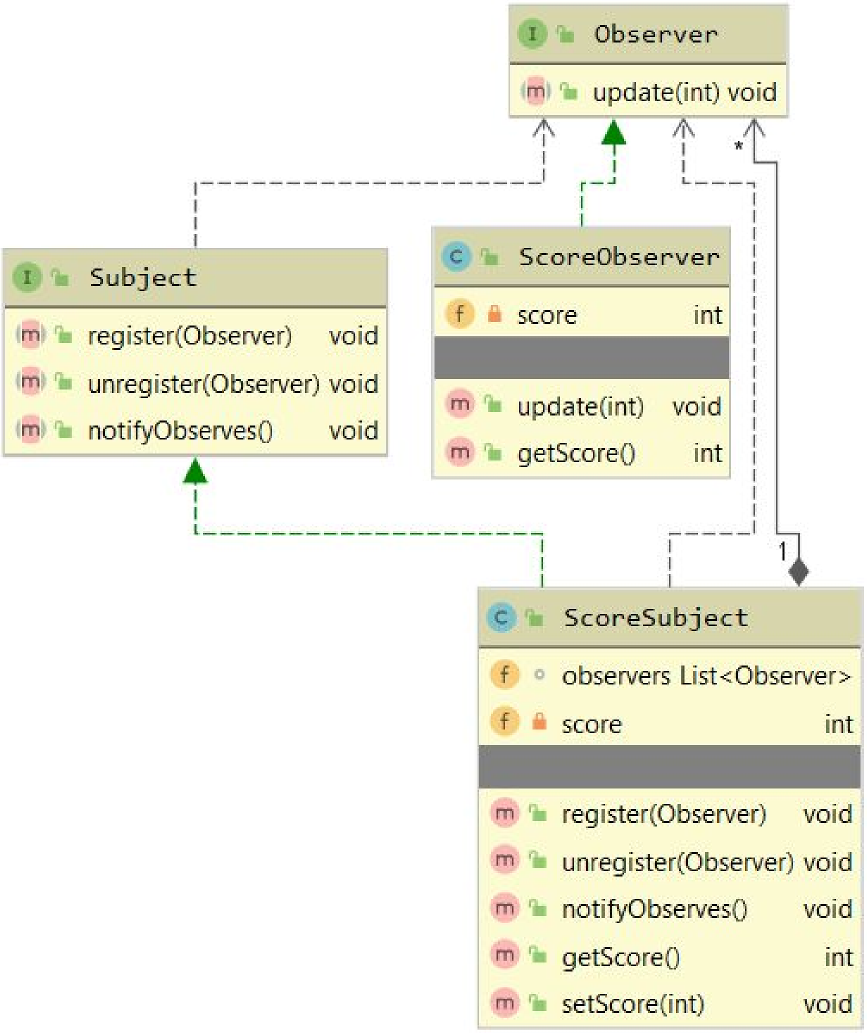


ObjectPool

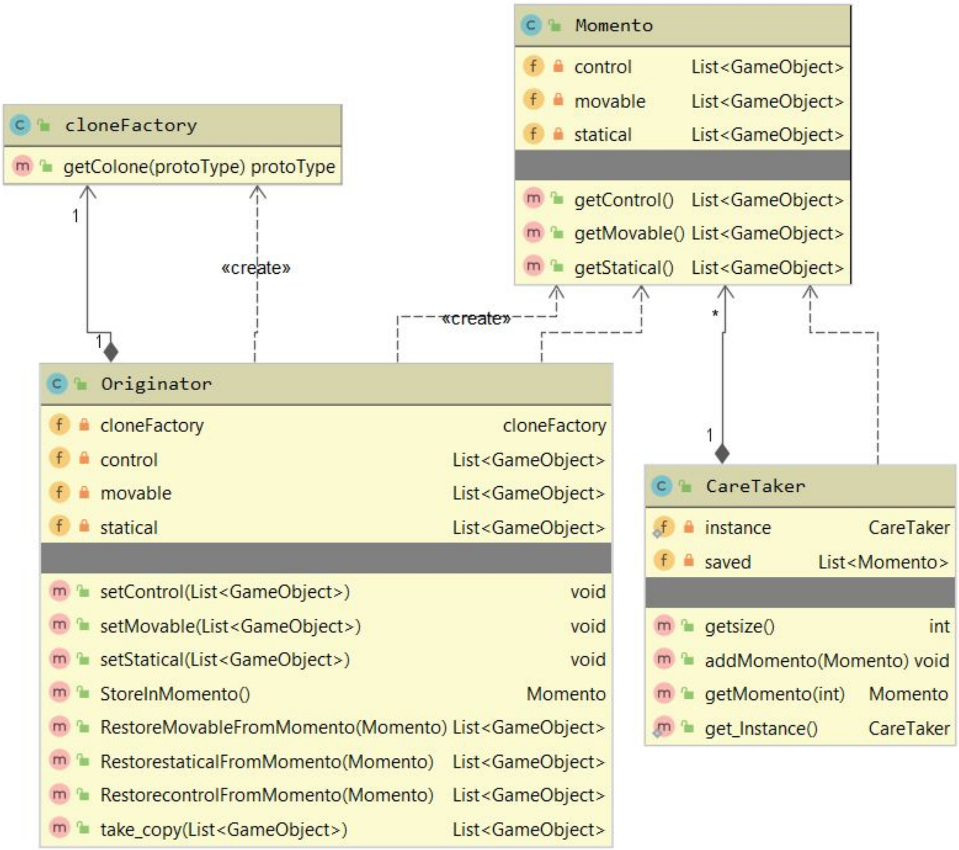
instance	ObjectPool
reusables	ArrayList<GameObject>
maxsize	int
world	World
plateFactory	PlateFactory
workSpace	workspace
random	Random
iterator	int
direct	int
getWorld()	World
setWorld(World)	void
setMaxsize(int)	void
aquireimage()	ArrayList<GameObject>
releaseImage(ImageObject)	void
get_instance()	ObjectPool

5-observer design pattern

Created for calculating the score

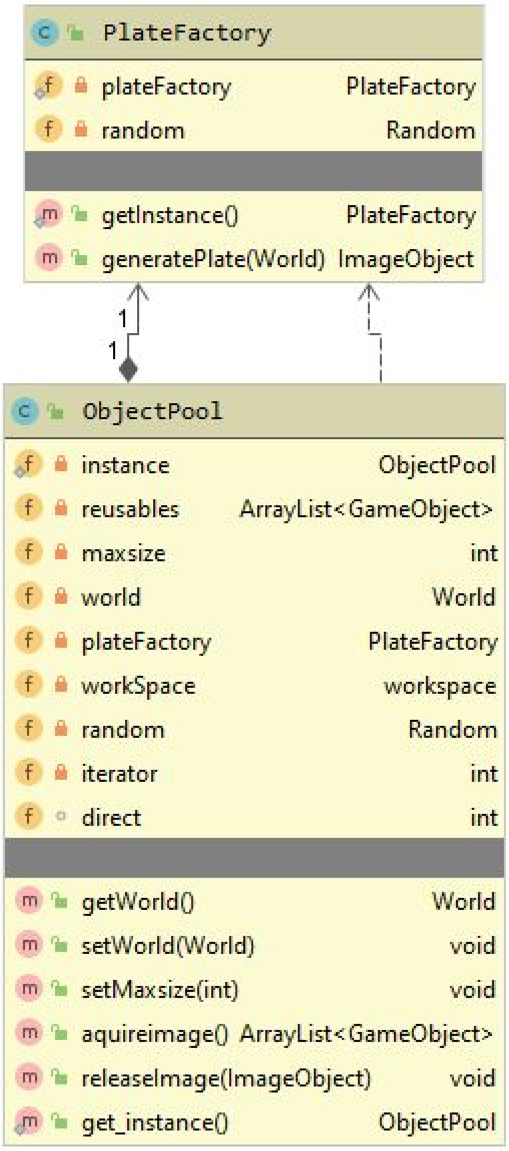


6-momento “snapshot” design pattern
Created for taking a copy for moving , static and controllable so we can replay the game if he want


Momento
control List<GameObject>
movable List<GameObject>
statical List<GameObject>
getControl() List<GameObject>
getMovable() List<GameObject>
getStatical() List<GameObject>
cloneFactory
getColone(protoType) protoType
1
«create»
«create»
1
*
1
Originator
cloneFactory cloneFactory
control List<GameObject>
movable List<GameObject>
statical List<GameObject>
setControl(List<GameObject>) void
setMovable(List<GameObject>) void
setStatical(List<GameObject>) void
StoreInMomento() Momento
RestoreMovableFromMomento(Momento) List<GameObject>
RestorestaticalFromMomento(Momento) List<GameObject>
RestorecontrolFromMomento(Momento) List<GameObject>
take_copy(List<GameObject>) List<GameObject>
CareTaker
instance CareTaker
saved List<Momento>
getsize() int
addMomento(Momento) void
getMomento(int) Momento
get_Instance() CareTaker

7-singleton design pattern

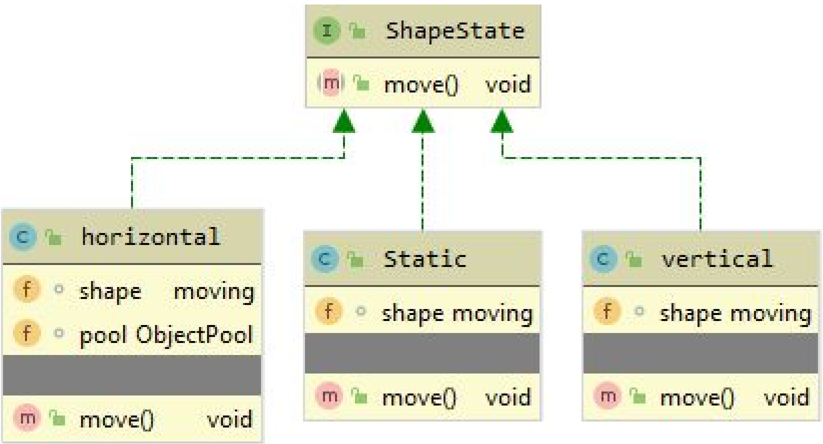
Created for making a single object pool and a single plate factory



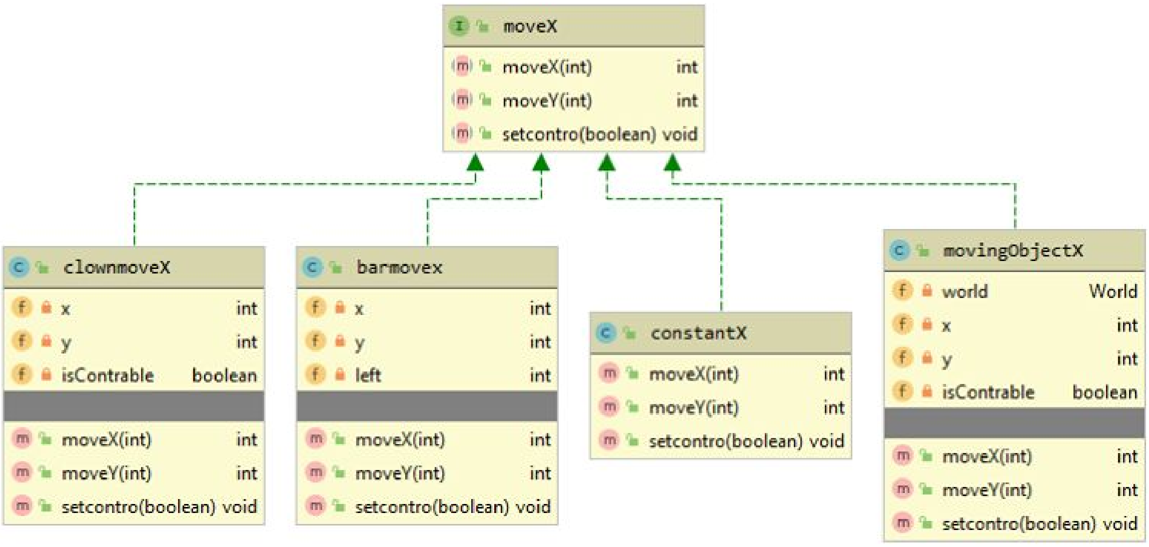
PlateFactory
plateFactory PlateFactory
random Random
getInstance() PlateFactory
generatePlate(World) ImageObject
1
1
ObjectPool
instance ObjectPool
reusables ArrayList<GameObject>
maxsize int
world World
plateFactory PlateFactory
workSpace workspace
random Random
iterator int
direct int
getWorld() World
setWorld(World) void
setMaxsize(int) void
aquireimage() ArrayList<GameObject>
releaseImage(ImageObject) void
get_instance() ObjectPool

8-state design pattern

Created for moving object that objects move vertical and horizontal

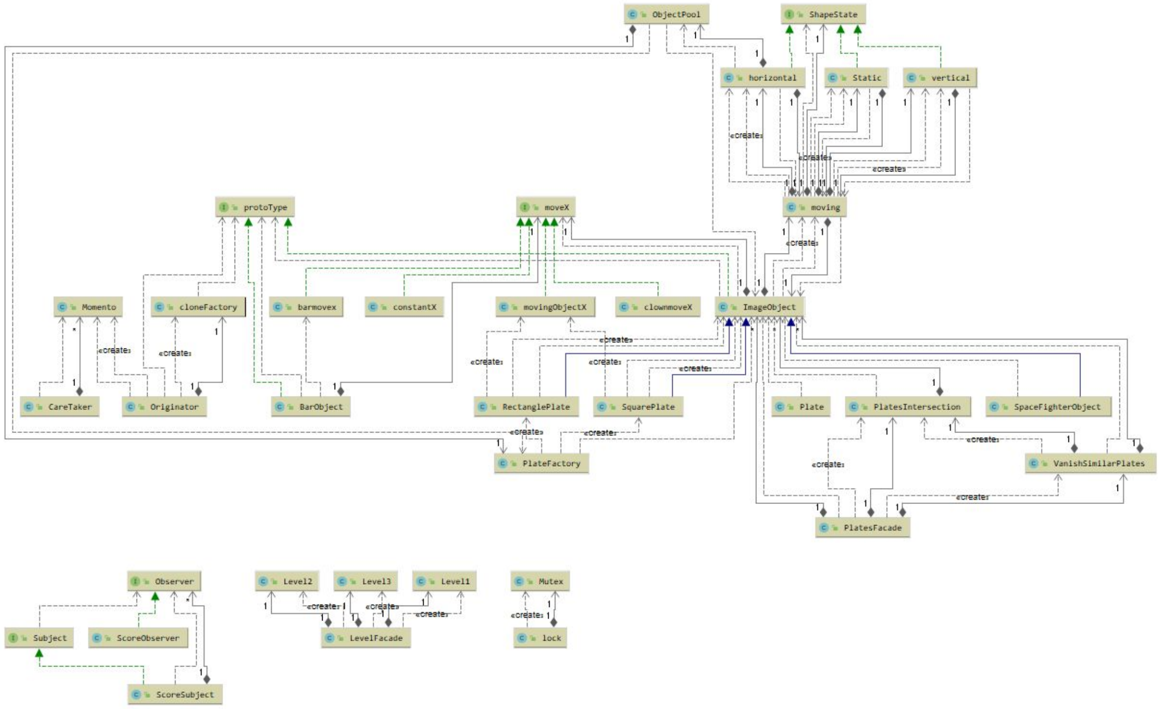


9- strategy design pattern
Created for moving in X direction to prevent moving object to intersect with each other



10 - iterator design pattern
Created for looping in moving object in refresh method

The whole uml without methods and attributes


ObjectPool
ShapeState
horizontal
Static
vertical
moving
protoType
moveX
Momento
cloneFactory
barmovex
constantX
movingObjectX
clownmoveX
ImageObject
CareTaker
Originator
BarObject
RectanglePlate
SquarePlate
Plate
PlatesIntersection
SpaceFighterObject
PlateFactory
VanishSimilarPlates
PlatesFacade
Observer
Subject
ScoreObserver
ScoreSubject
Level2
Level3
Level1
LevelFacade
Mutex
lock
«create»
1
*