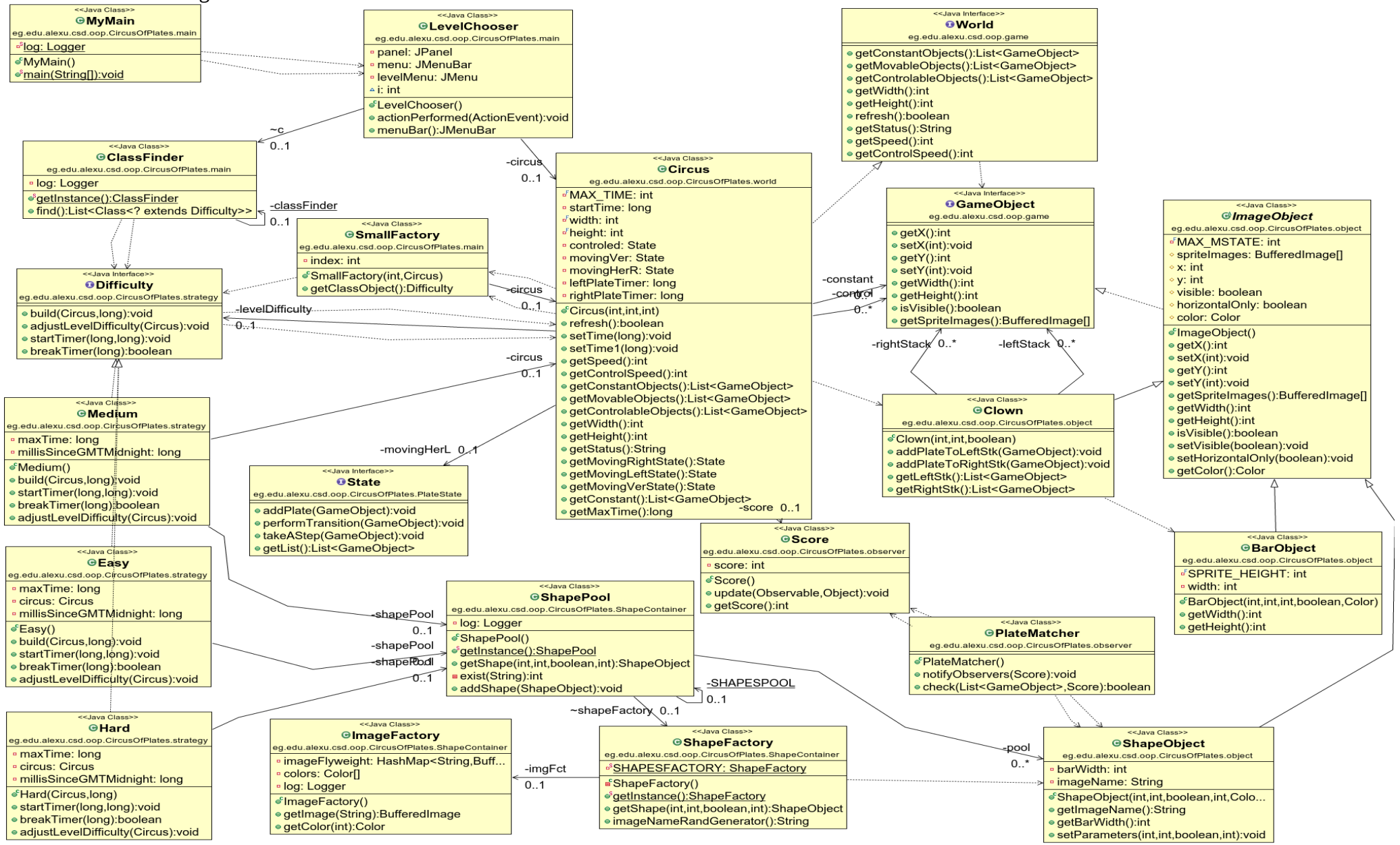
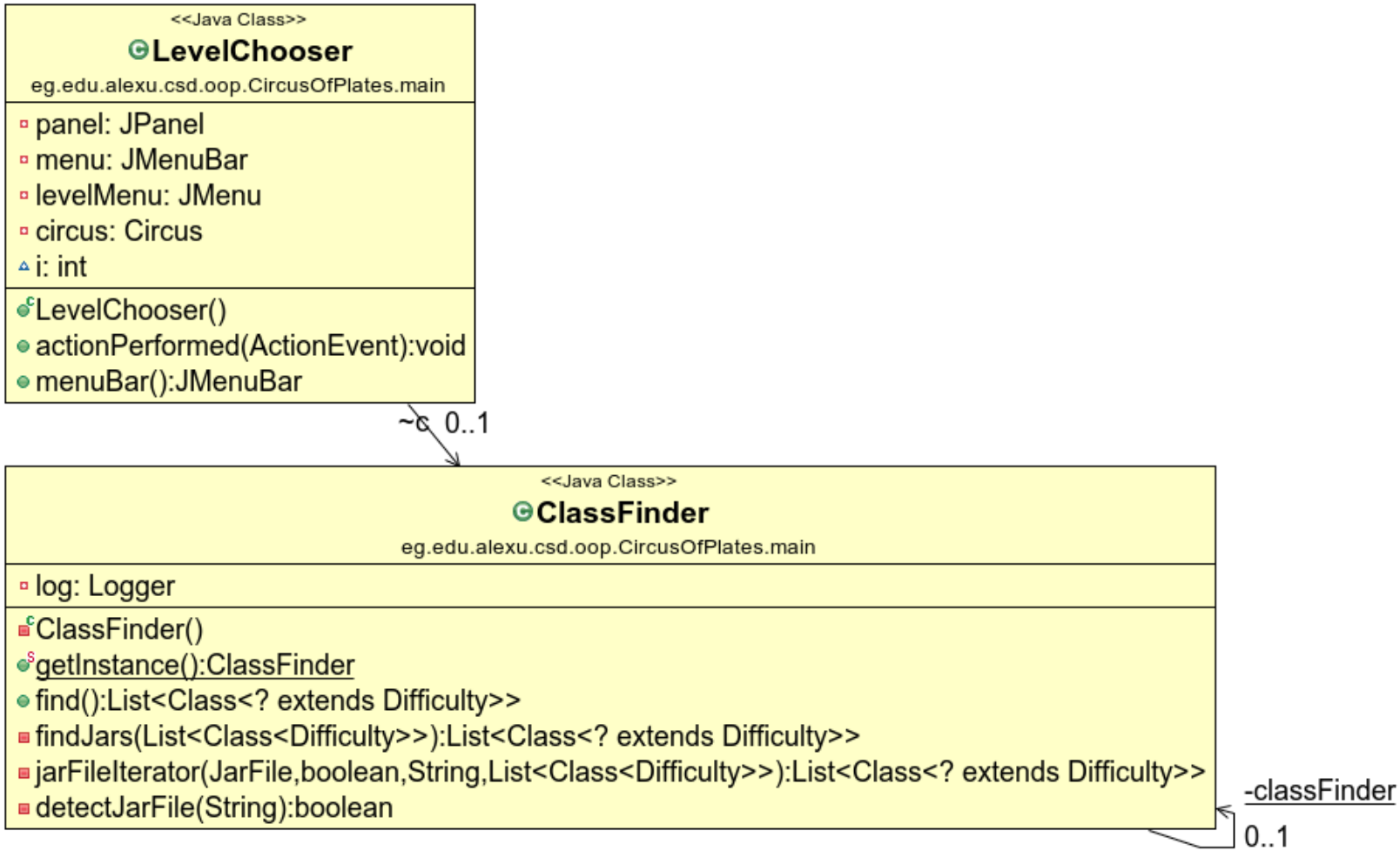
Circus Of Plates Game Report

Overall class UML diagram:

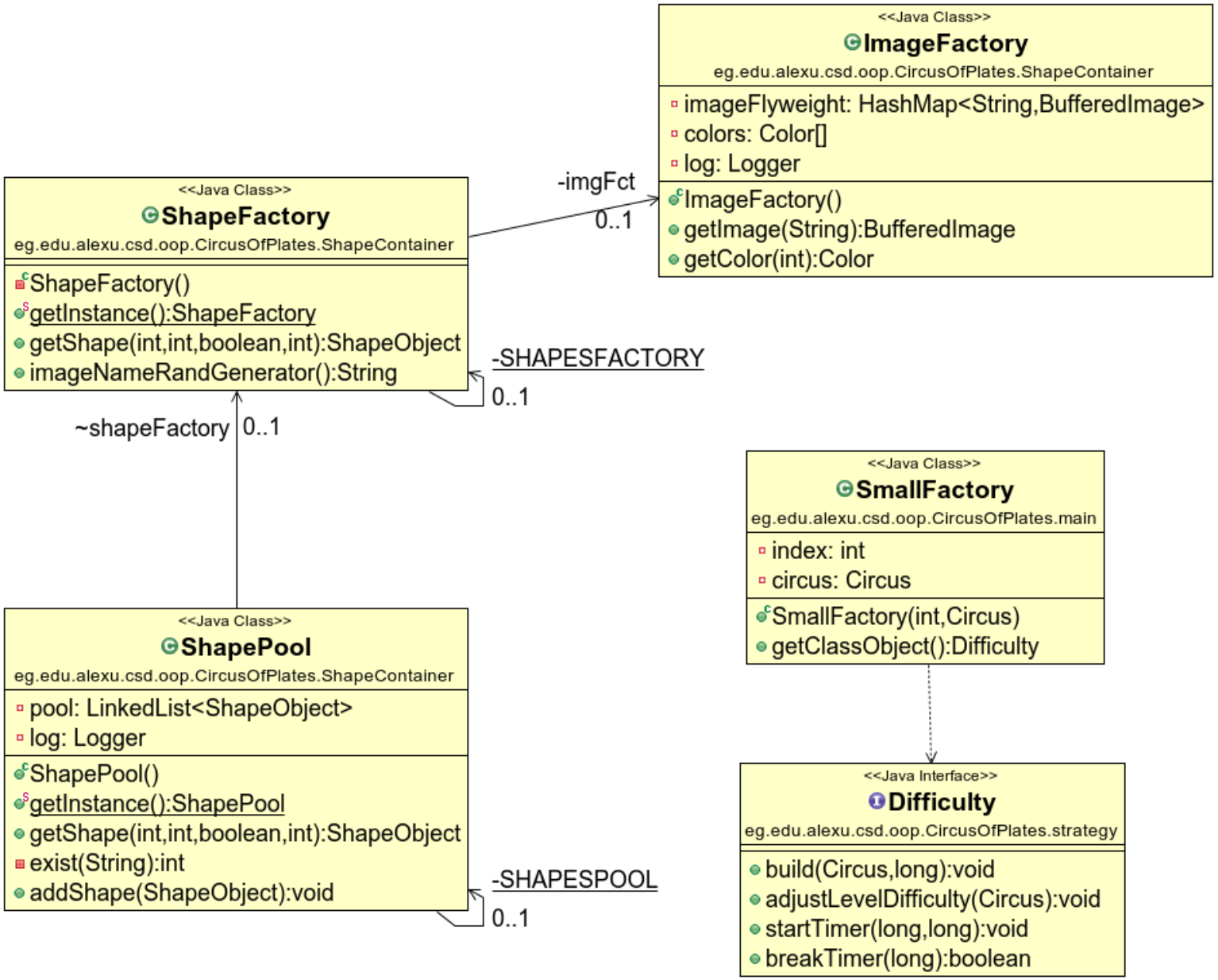


DESIGN PATTERN UML DIAGRAMS:

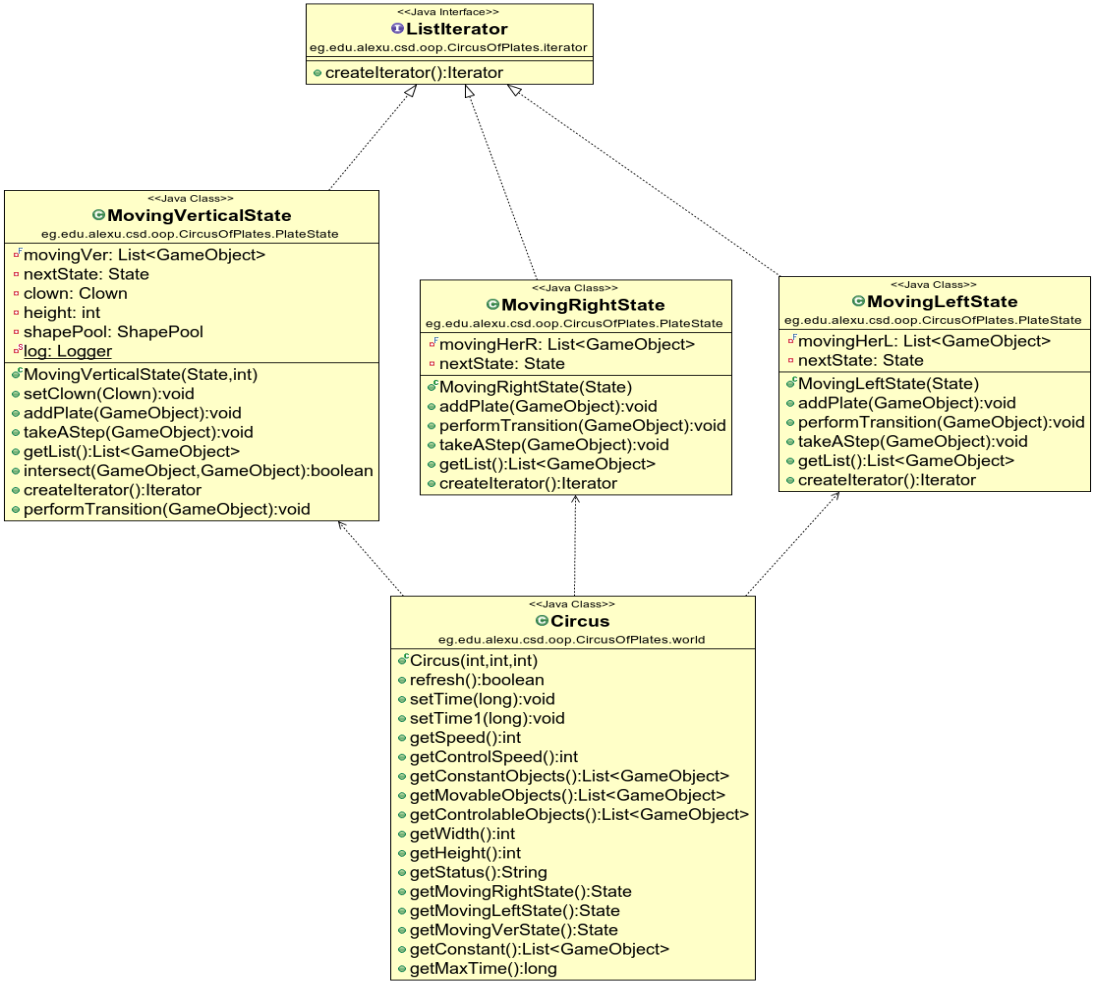
* Dynamic Linkage



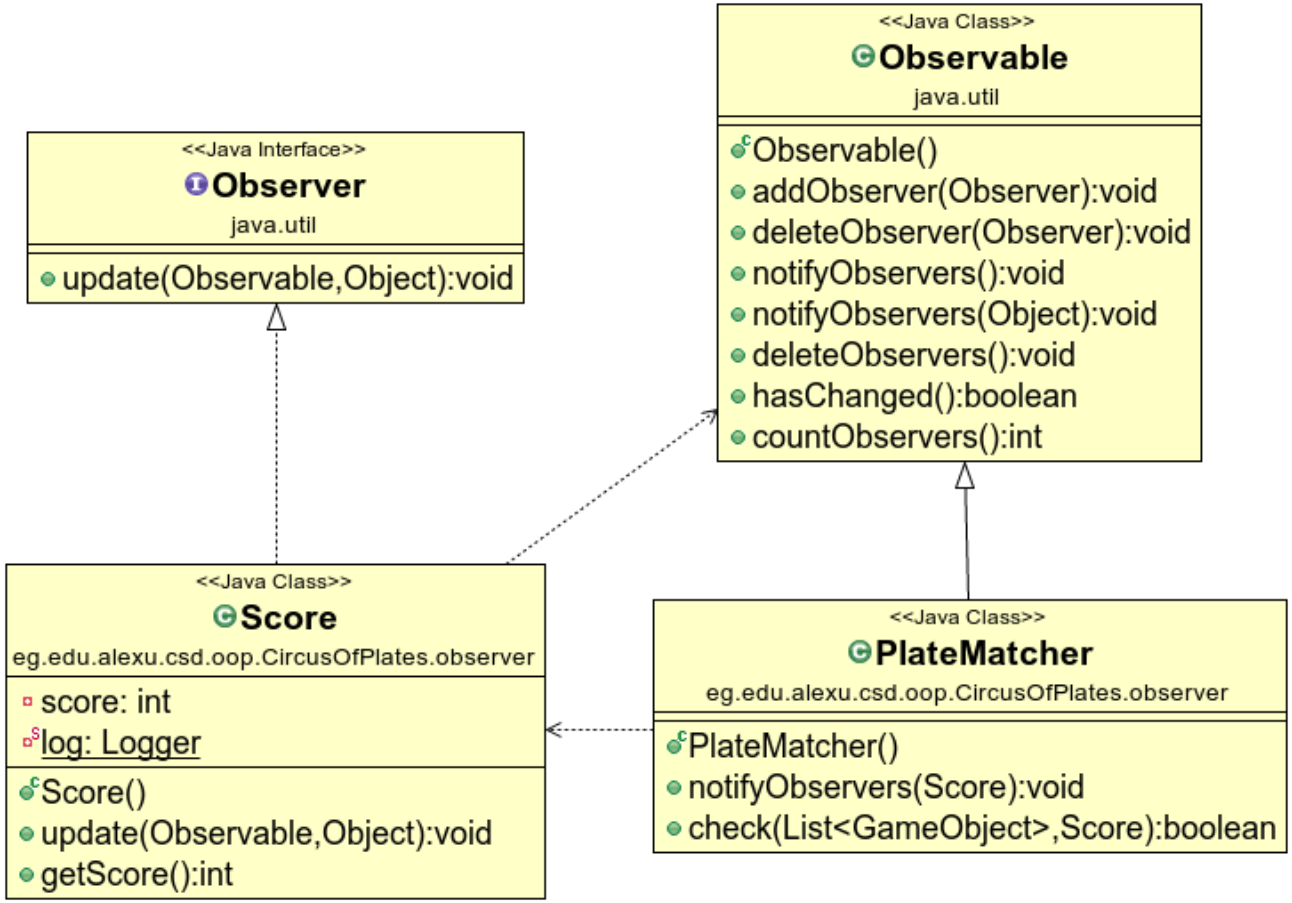
* Factory, Object pool and Flyweight:



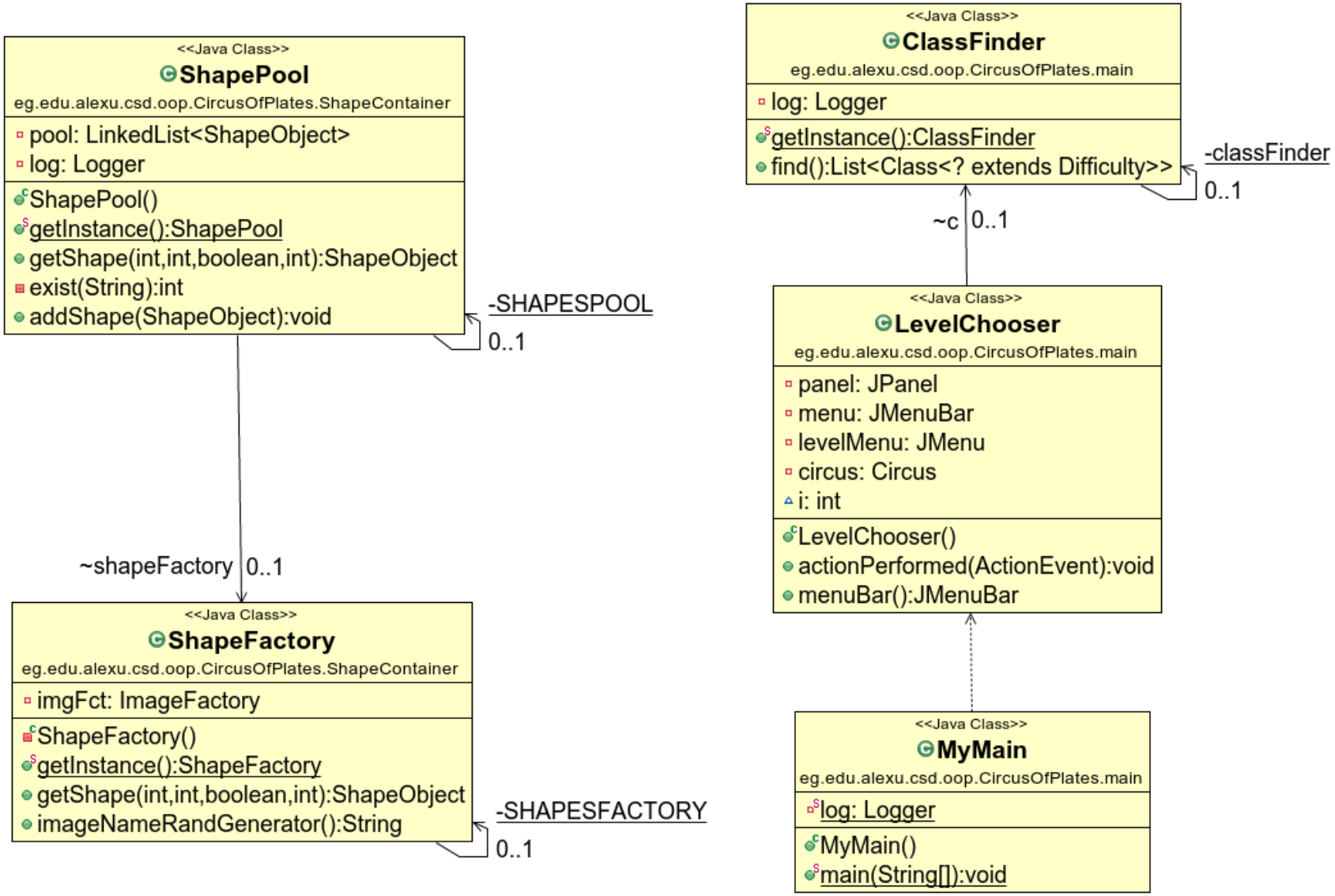
* Iterator:


<<Java Interface>>
ListIterator
eg.edu.alexu.csd.oop.CircusOfPlates.iterator
createIterator():Iterator
<<Java Class>>
MovingVerticalState
eg.edu.alexu.csd.oop.CircusOfPlates.PlateState
movingVer: List<GameObject>
nextState: State
clown: Clown
height: int
shapePool: ShapePool
log: Logger
MovingVerticalState(State,int)
setClown(Clown):void
addPlate(GameObject):void
takeAStep(GameObject):void
getList():List<GameObject>
intersect(GameObject,GameObject):boolean
createIterator():Iterator
performTransition(GameObject):void
<<Java Class>>
MovingRightState
eg.edu.alexu.csd.oop.CircusOfPlates.PlateState
movingHerR: List<GameObject>
nextState: State
MovingRightState(State)
addPlate(GameObject):void
performTransition(GameObject):void
takeAStep(GameObject):void
getList():List<GameObject>
createIterator():Iterator
<<Java Class>>
MovingLeftState
eg.edu.alexu.csd.oop.CircusOfPlates.PlateState
movingHerL: List<GameObject>
nextState: State
MovingLeftState(State)
addPlate(GameObject):void
performTransition(GameObject):void
takeAStep(GameObject):void
getList():List<GameObject>
createIterator():Iterator
<<Java Class>>
Circus
eg.edu.alexu.csd.oop.CircusOfPlates.world
Circus(int,int,int)
refresh():boolean
setTime(long):void
setTime1(long):void
getSpeed():int
getControlSpeed():int
getConstantObjects():List<GameObject>
getMovableObjects():List<GameObject>
getControlableObjects():List<GameObject>
getWidth():int
getHeight():int
getStatus():String
getMovingRightState():State
getMovingLeftState():State
getMovingVerState():State
getConstant():List<GameObject>
getMaxTime():long

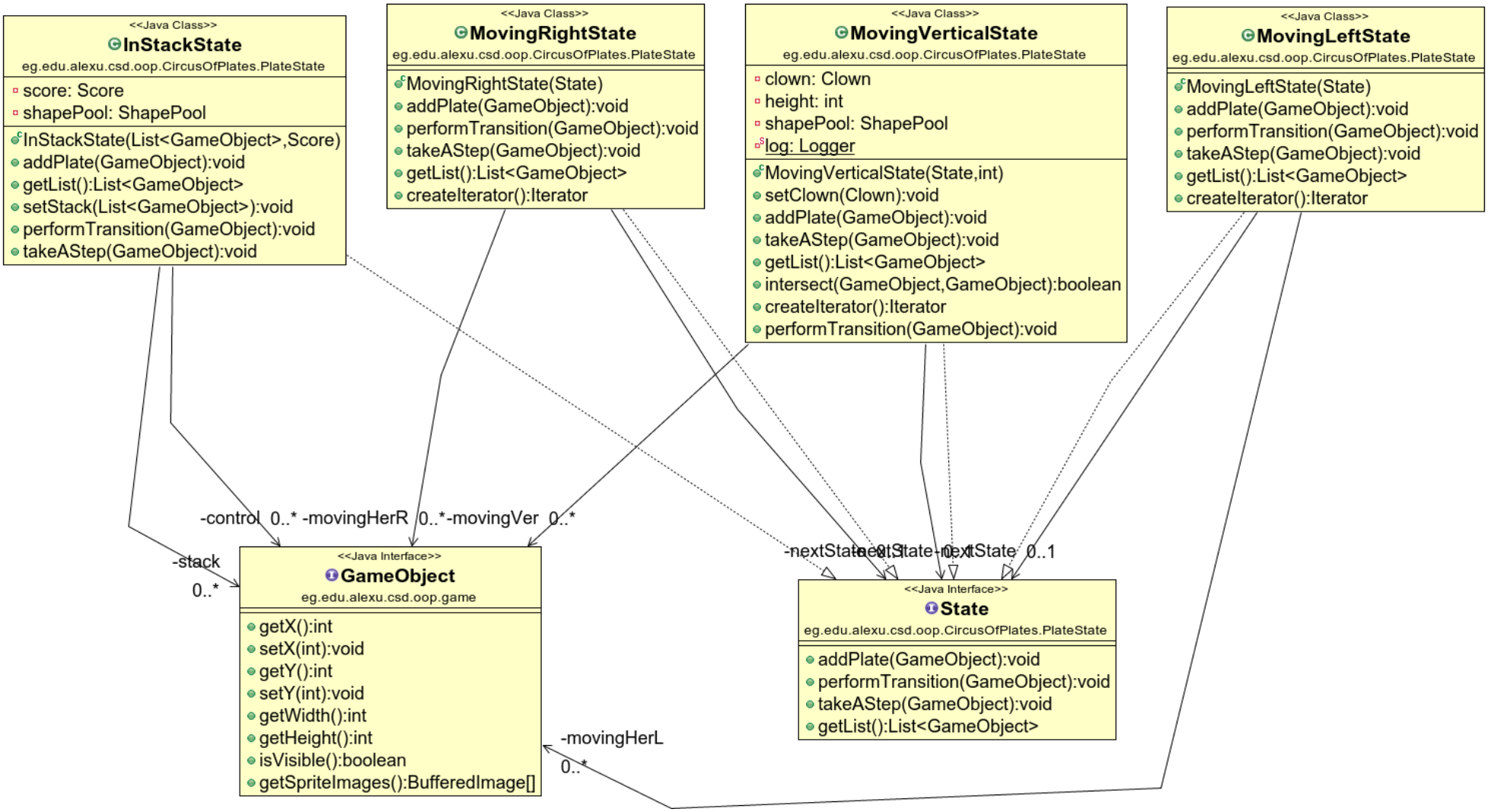
* Observer :


<<Java Class>>
Observable
java.util
Observable()
addObserver(Observer):void
deleteObserver(Observer):void
notifyObservers():void
notifyObservers(Object):void
deleteObservers():void
hasChanged():boolean
countObservers():int
<<Java Interface>>
Observer
java.util
update(Observable,Object):void
<<Java Class>>
Score
eg.edu.alexu.csd.oop.CircusOfPlates.observer
score: int
log: Logger
Score()
update(Observable,Object):void
getScore():int
<<Java Class>>
PlateMatcher
eg.edu.alexu.csd.oop.CircusOfPlates.observer
PlateMatcher()
notifyObservers(Score):void
check(List<GameObject>,Score):boolean

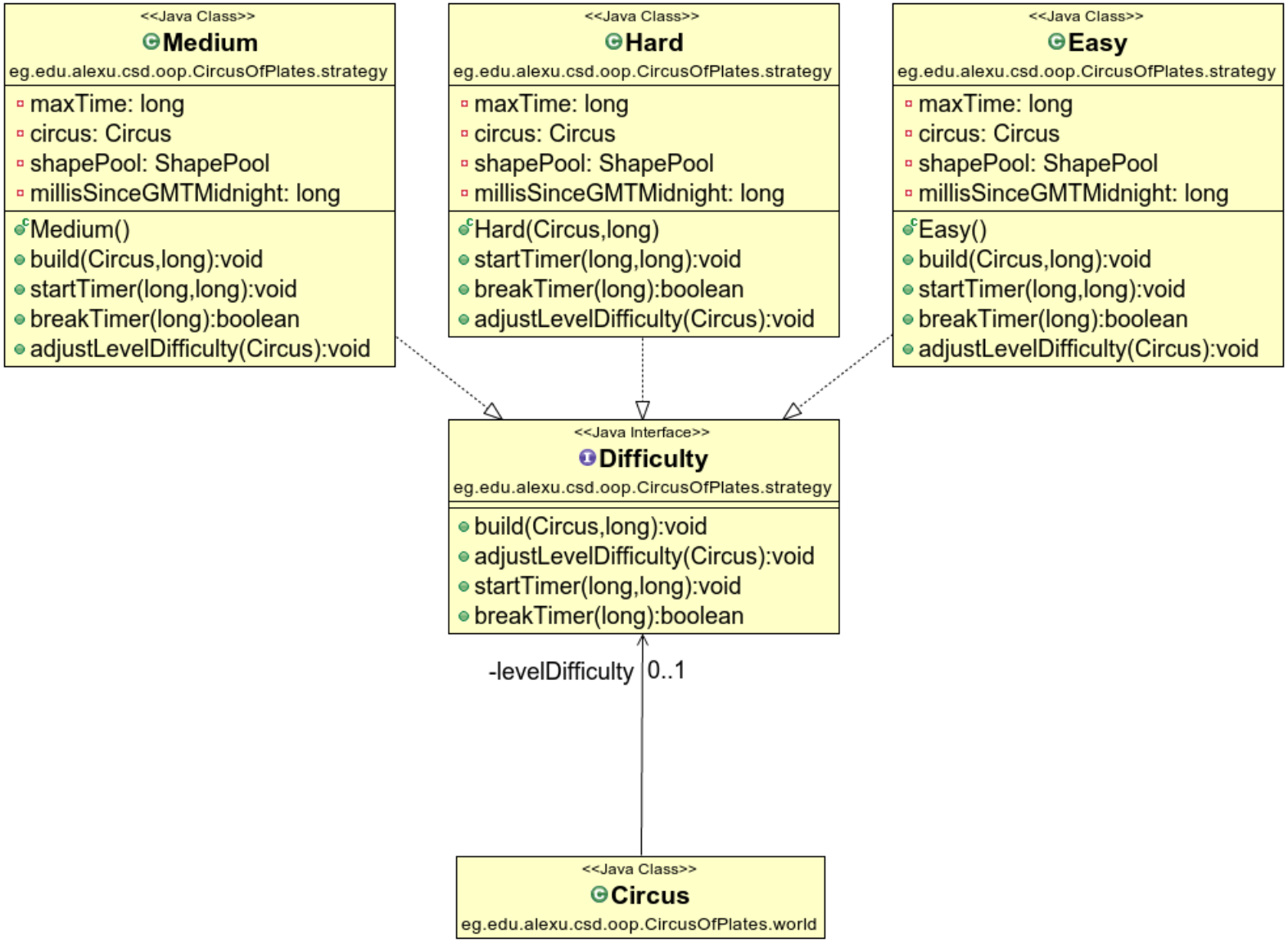
* Singleton:


<<Java Class>>
ShapePool
eg.edu.alexu.csd.oop.CircusOfPlates.ShapeContainer
pool: LinkedList<ShapeObject>
log: Logger
ShapePool()
getInstance():ShapePool
getShape(int,int,boolean,int):ShapeObject
exist(String):int
addShape(ShapeObject):void
-SHAPESPOOL
0..1
~shapeFactory
0..1
<<Java Class>>
ShapeFactory
eg.edu.alexu.csd.oop.CircusOfPlates.ShapeContainer
imgFct: ImageFactory
ShapeFactory()
getInstance():ShapeFactory
getShape(int,int,boolean,int):ShapeObject
imageNameRandGenerator():String
-SHAPESFACTORY
0..1
<<Java Class>>
ClassFinder
eg.edu.alexu.csd.oop.CircusOfPlates.main
log: Logger
getInstance():ClassFinder
find():List<Class<? extends Difficulty>>
-classFinder
0..1
~c
0..1
<<Java Class>>
LevelChooser
eg.edu.alexu.csd.oop.CircusOfPlates.main
panel: JPanel
menu: JMenuBar
levelMenu: JMenu
circus: Circus
i: int
LevelChooser()
actionPerformed(ActionEvent):void
menuBar():JMenuBar
<<Java Class>>
MyMain
eg.edu.alexu.csd.oop.CircusOfPlates.main
log: Logger
MyMain()
main(String[]):void

* State:


<<Java Class>>
InStackState
eg.edu.alexu.csd.oop.CircusOfPlates.PlateState
score: Score
shapePool: ShapePool
InStackState(List<GameObject>,Score)
addPlate(GameObject):void
getList():List<GameObject>
setStack(List<GameObject>):void
performTransition(GameObject):void
takeAStep(GameObject):void
<<Java Class>>
MovingRightState
eg.edu.alexu.csd.oop.CircusOfPlates.PlateState
MovingRightState(State)
addPlate(GameObject):void
performTransition(GameObject):void
takeAStep(GameObject):void
getList():List<GameObject>
createIterator():Iterator
<<Java Class>>
MovingVerticalState
eg.edu.alexu.csd.oop.CircusOfPlates.PlateState
clown: Clown
height: int
shapePool: ShapePool
log: Logger
MovingVerticalState(State,int)
setClown(Clown):void
addPlate(GameObject):void
takeAStep(GameObject):void
getList():List<GameObject>
intersect(GameObject,GameObject):boolean
createIterator():Iterator
performTransition(GameObject):void
<<Java Class>>
MovingLeftState
eg.edu.alexu.csd.oop.CircusOfPlates.PlateState
MovingLeftState(State)
addPlate(GameObject):void
performTransition(GameObject):void
takeAStep(GameObject):void
getList():List<GameObject>
createIterator():Iterator
-control 0..*
-movingHerR 0..*
-movingVer 0..*
-stack
0..*
-nextState 0..1
-nextState 0..1
-nextState 0..1
<<Java Interface>>
GameObject
eg.edu.alexu.csd.oop.game
getX():int
setX(int):void
getY():int
setY(int):void
getWidth():int
getHeight():int
isVisible():boolean
getSpriteImages():BufferedImage[]
-movingHerL
0..*
<<Java Interface>>
State
eg.edu.alexu.csd.oop.CircusOfPlates.PlateState
addPlate(GameObject):void
performTransition(GameObject):void
takeAStep(GameObject):void
getList():List<GameObject>

* Strategy:


<<Java Class>>
Medium
eg.edu.alexu.csd.oop.CircusOfPlates.strategy
maxTime: long
circus: Circus
shapePool: ShapePool
millisSinceGMTMidnight: long
Medium()
build(Circus,long):void
startTimer(long,long):void
breakTimer(long):boolean
adjustLevelDifficulty(Circus):void
<<Java Class>>
Hard
eg.edu.alexu.csd.oop.CircusOfPlates.strategy
maxTime: long
circus: Circus
shapePool: ShapePool
millisSinceGMTMidnight: long
Hard(Circus,long)
startTimer(long,long):void
breakTimer(long):boolean
adjustLevelDifficulty(Circus):void
<<Java Class>>
Easy
eg.edu.alexu.csd.oop.CircusOfPlates.strategy
maxTime: long
circus: Circus
shapePool: ShapePool
millisSinceGMTMidnight: long
Easy()
build(Circus,long):void
startTimer(long,long):void
breakTimer(long):boolean
adjustLevelDifficulty(Circus):void
<<Java Interface>>
Difficulty
eg.edu.alexu.csd.oop.CircusOfPlates.strategy
build(Circus,long):void
adjustLevelDifficulty(Circus):void
startTimer(long,long):void
breakTimer(long):boolean
-levelDifficulty
0..1
<<Java Class>>
Circus
eg.edu.alexu.csd.oop.CircusOfPlates.world