

World of Plastics

DOCTOR

Dr. Khaled Nagy

TAS

Eng. Shehab K. El-semmany

Eng. Paula B. Bassily

STUDENTS

Abanoub Ashraaf Ezzat (01)

Arsany Atef Abdo (08)

Kirellos Malak Habib (33)

Michael Said Beshara (36)

Description:

It is single player-game in which each clown carries two stacks of plates, and there are set of colored plates queues that fall and he tries to catch them, if he manages to collect three consecutive plates of the same color, then they are vanished and his score increases. You are free to put rules ending the game.



Design Decisions:

- Choosing difficulty and mode is by enum which can be changed according to the user's choices.
- Loading game objects is dynamically according to the mode.
- Each functionality has its own package which contains all needed handling.
- There are 18 different packages contain the different code parts.
- Pool design pattern is used to create falling objects.
- During running the game changes done by using the menu bar.
- Snapshot is taken on saving the game by saving each used variable beside the user name into a Json file.
- Factory collects game objects and create game mode.
- Fly wait take the unused fallen objects and return them again to the pool.

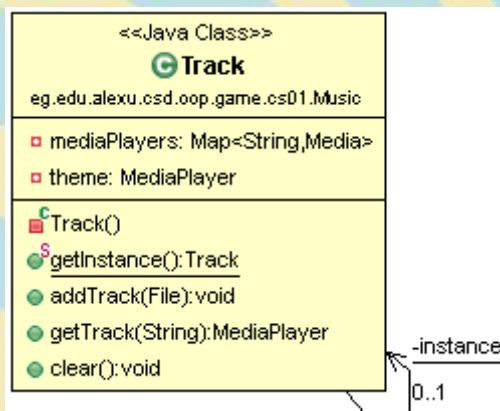
- Thread plays music in the background of the game.

Design Patterns used and their UML

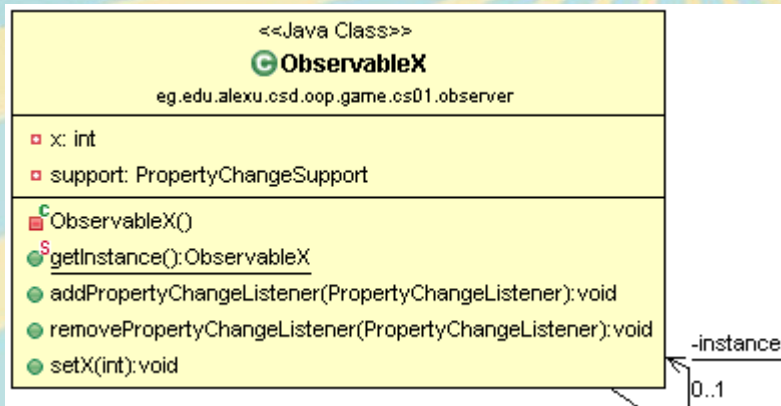
Class Diagrams:

Singleton

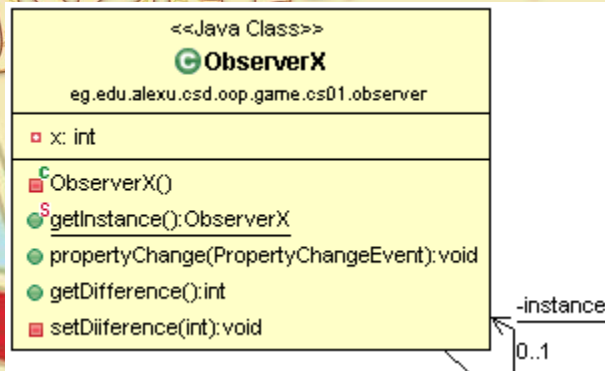
Track



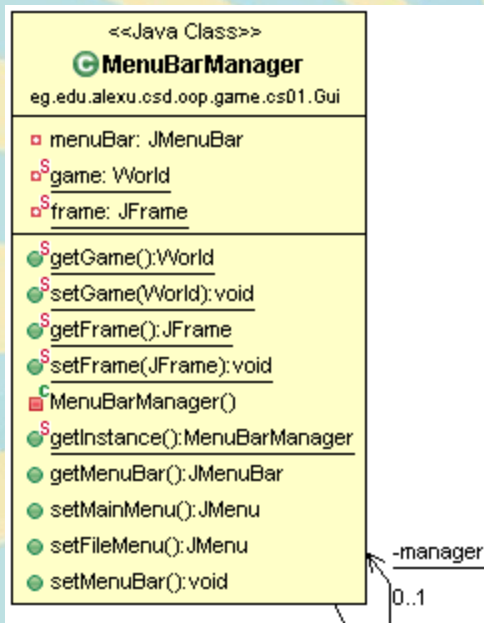
ObservableX



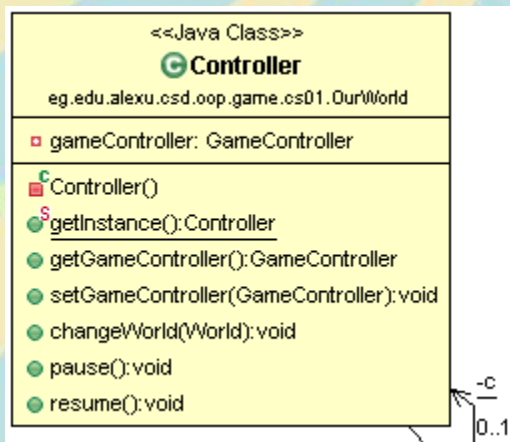
ObserverX








MenuBarManager











Controller



Snapshot

<<Java Class>>	
 Snapshot	
eg.edu.alexu.csd.oop.game.cs01.SnapShot	
	Snapshot()
	getSnapShot(): SnapShot
	saveGame(World,String): void
	loadGame(String): World
-snapShot	
0..1	

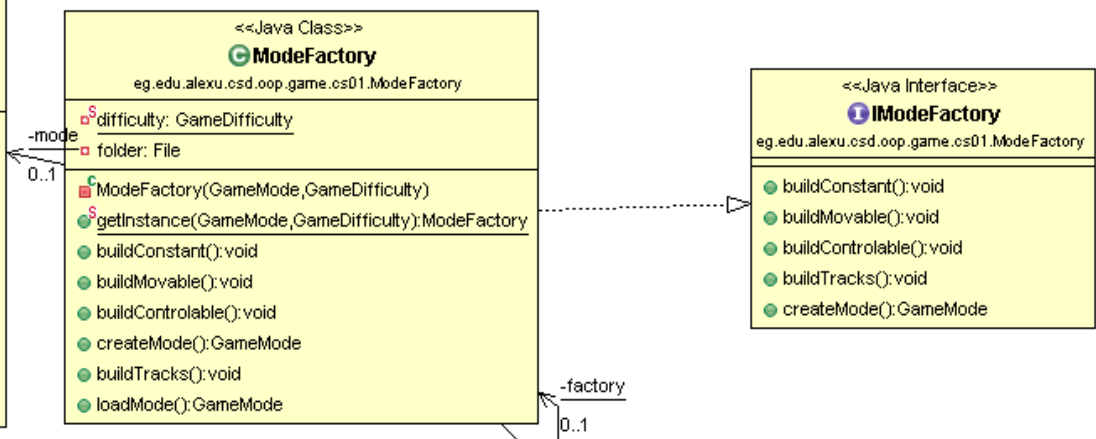
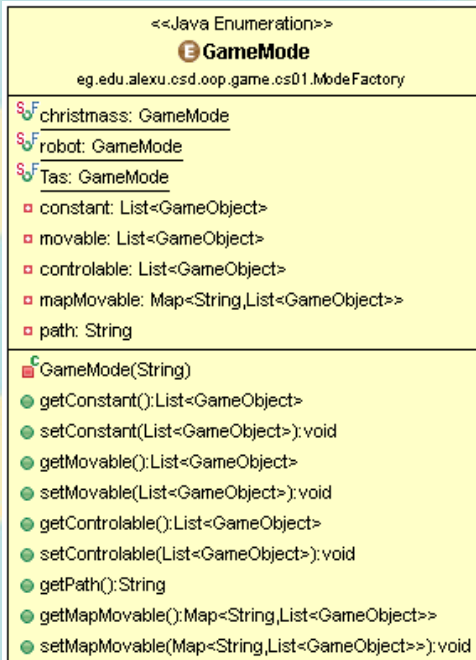
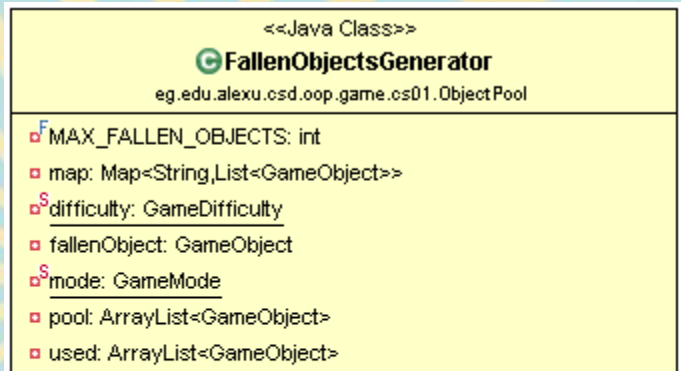
GameObjectLoader

<<Java Class>>	
 GameObjectLoader	
eg.edu.alexu.csd.oop.game.cs01.DynamicLinkage	
□ classesMap: Map<String,Class<?>>	
	GameObjectLoader()
	getInstance(): GameObjectLoader
	loadClass(String): Class<?>
	getClassesMap(): Map<String,Class<?>>
	setClassesMap(Map<String,Class<?>>): void
	loadClasses(): void
	addClass(String,String): void
-instance	
0..1	

World of plates

Factory
ModeFactory
IModeFactory
GameMode

Pool
FallenObjectsGenerator

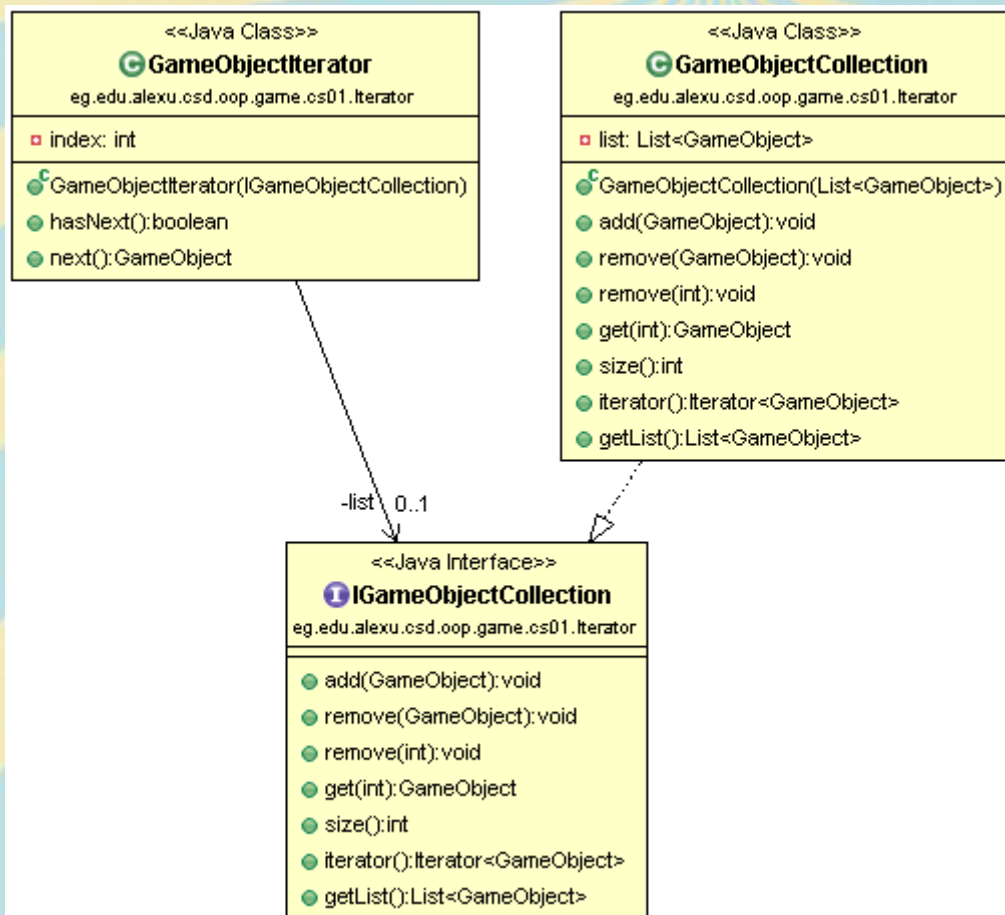


World of platées

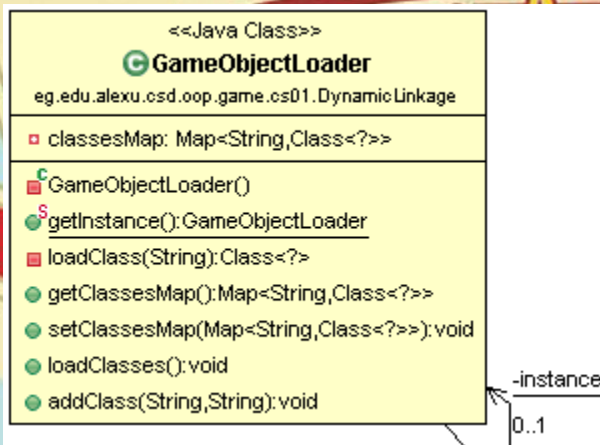
Iterator

GameObjectCollection

GameObjectIterator



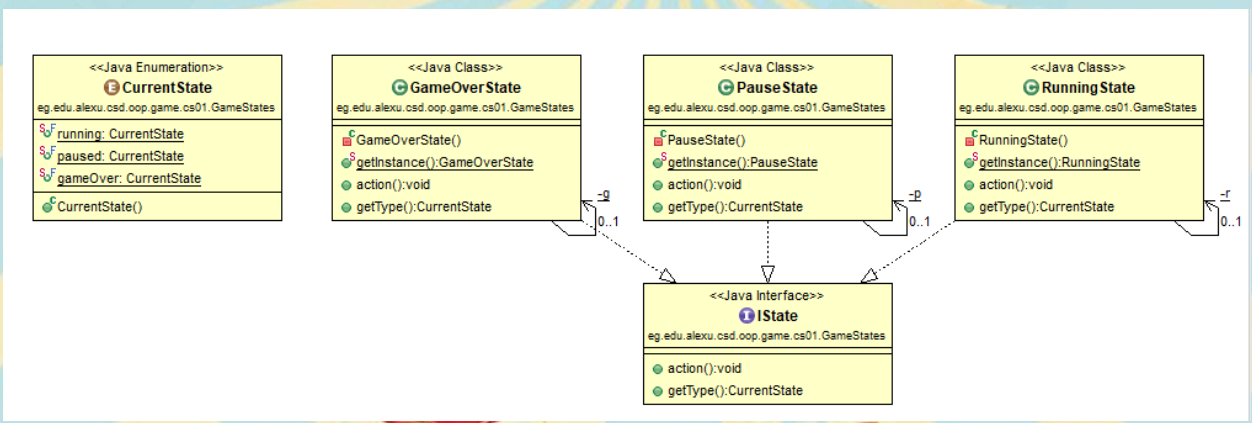
Dynamic Linkage GameObjectLoader



Used to load object classes dynamically during game run.

State

CurrentState



Fly Weight

All of the game objects have a reference to an image according to its properties, to avoid having much of redundant unused memory.

World of platces

Snap Shot

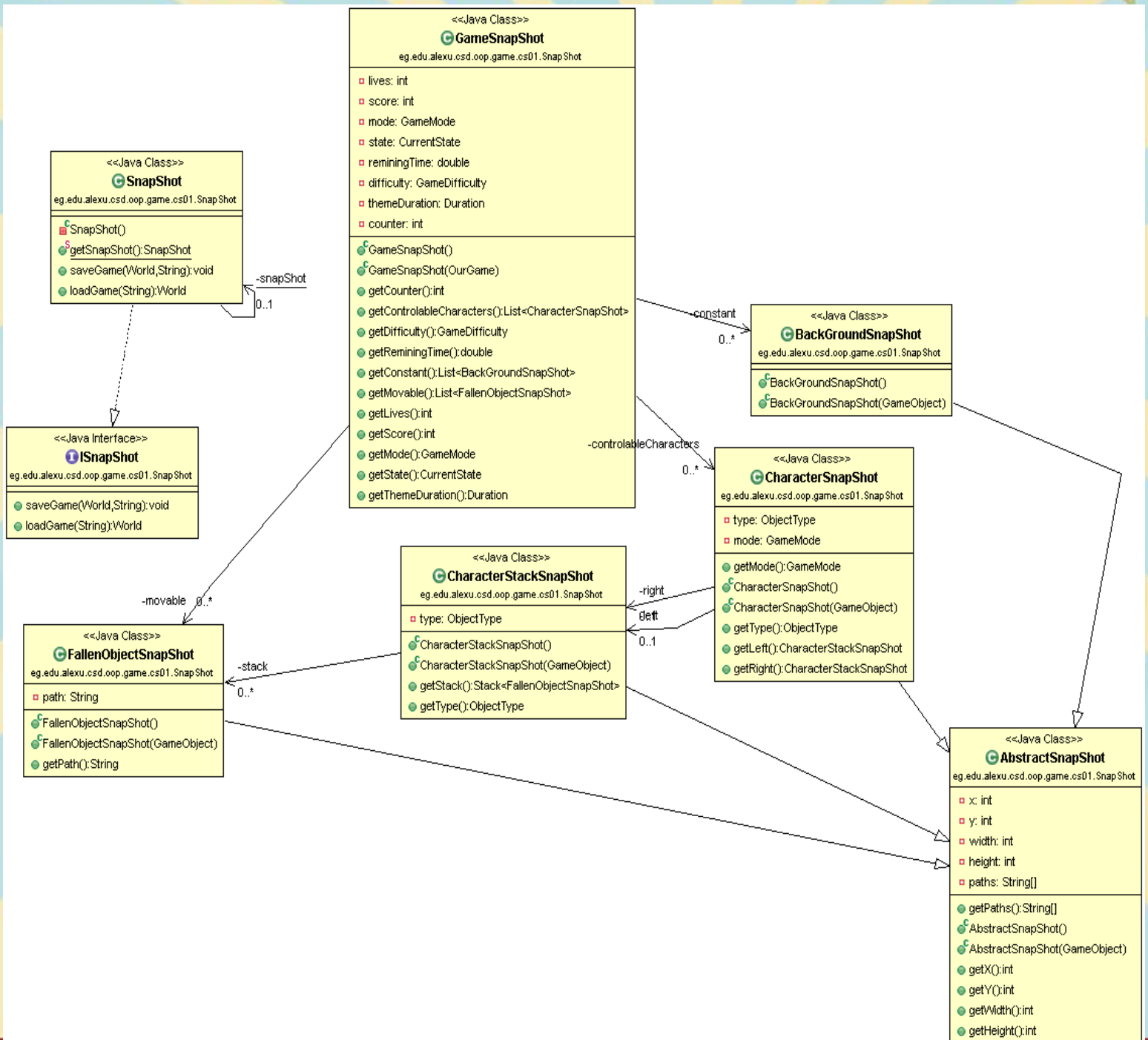
BackGroundSnapShot

CharacterSnapShot

CharacterStackSnapShot

FallenObjectsSnapShot

GameSnapShotSnapShot



World of platces

Strategy

MovableX

MovableXCondition

MovableY

NotMovableY

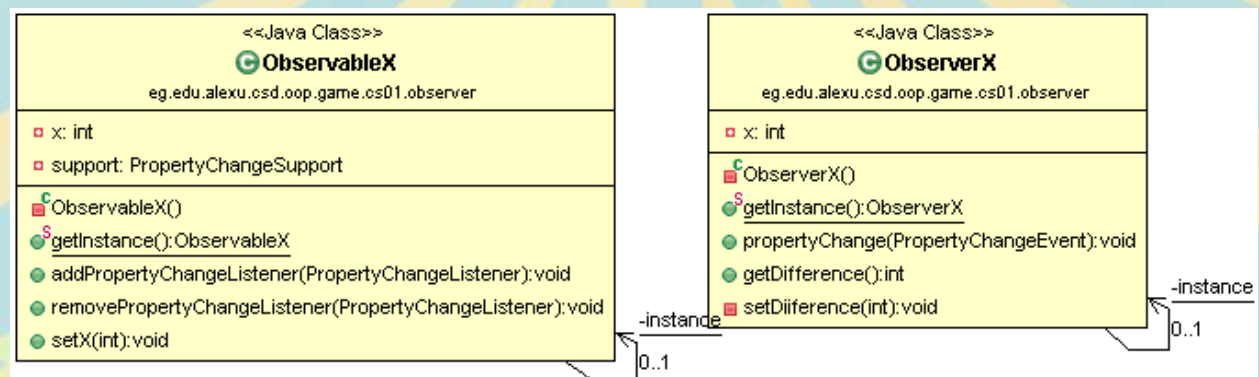
NotMovableX



Observer

ObservableX

ObserverX



Immutable

Occurs in the SnapShot Pattern.

Proto Type

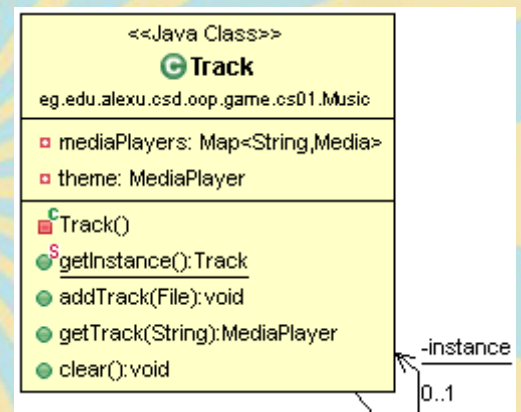
Used in cloning falling objects.

MVC

Used in connecting between backend models and view frontend.

Thread

It plays music parallel while the game is running.



Façade

It makes the game manager an abstraction for all the interior classes. All the interior classes are not seen outside the game manager

World of platies

Interface

IMovableX

IMovableY

```
<<Java Interface>>
IIMovableX
eg.edu.alexu.csd.oop.game.cs01.Strategy

• setX(int):int
```

```
<<Java Interface>>
IIMovableY
eg.edu.alexu.csd.oop.game.cs01.Strategy

• setY(int):int
```

Interface & Abstract Class

GameObjectCollection

IGameObjectCollection

```
<<Java Class>>
GGameObjectCollection
eg.edu.alexu.csd.oop.game.cs01.Iterator

▪ list: List<GameObject>

• GGameObjectCollection(List<GameObject>)
• add(GameObject):void
• remove(GameObject):void
• remove(int):void
• get(int):GameObject
• size():int
• iterator():Iterator<GameObject>
• getList():List<GameObject>
```

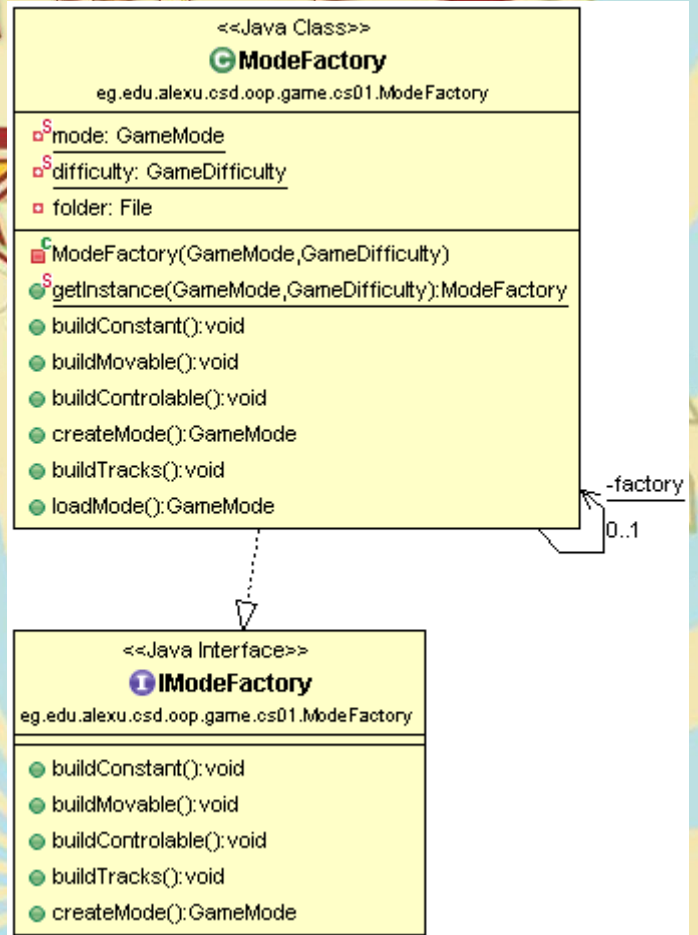
```
<<Java Interface>>
IIGameObjectCollection
eg.edu.alexu.csd.oop.game.cs01.Iterator

• add(GameObject):void
• remove(GameObject):void
• remove(int):void
• get(int):GameObject
• size():int
• iterator():Iterator<GameObject>
• getList():List<GameObject>
```


World of

ModeFactory

IModeFactory



AbstractGameObject

GameObject



OurGame World

<<Java Interface>>

IWorld

eg.edu.alexu.csd.oop.game

- getConstantObjects(): List<GameObject>
- getMovableObjects(): List<GameObject>
- getControlableObjects(): List<GameObject>
- getWidth(): int
- getHeight(): int
- refresh(): boolean
- getStatus(): String
- getSpeed(): int
- getControlSpeed(): int

<<Java Class>>

OurGame

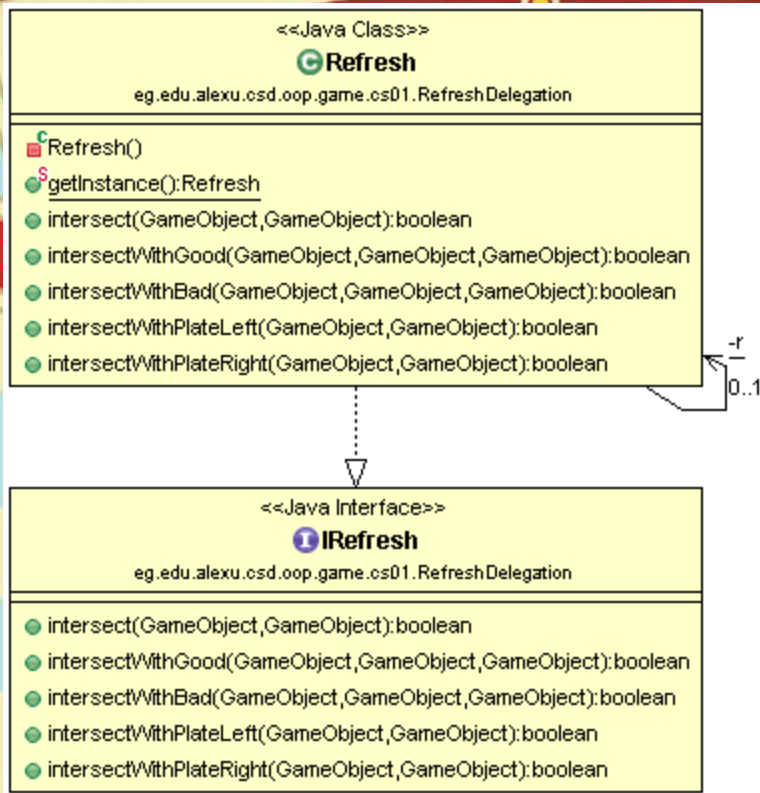
eg.edu.alexu.csd.oop.game.cs01.OurWorld

- ▣ ^SMAX_TIME: int
- ▣ lives: int
- ▣ score: int
- ▣ startTime: long
- ▣ constant: IGameObjectCollection
- ▣ movable: IGameObjectCollection
- ▣ controlable: IGameObjectCollection
- ▣ difficulty: GameDifficulty
- ▣ mode: GameMode
- ▣ state: CurrentState
- ▣ counter: int
- ▣ remainingTime: double
- ▣ duration: Duration

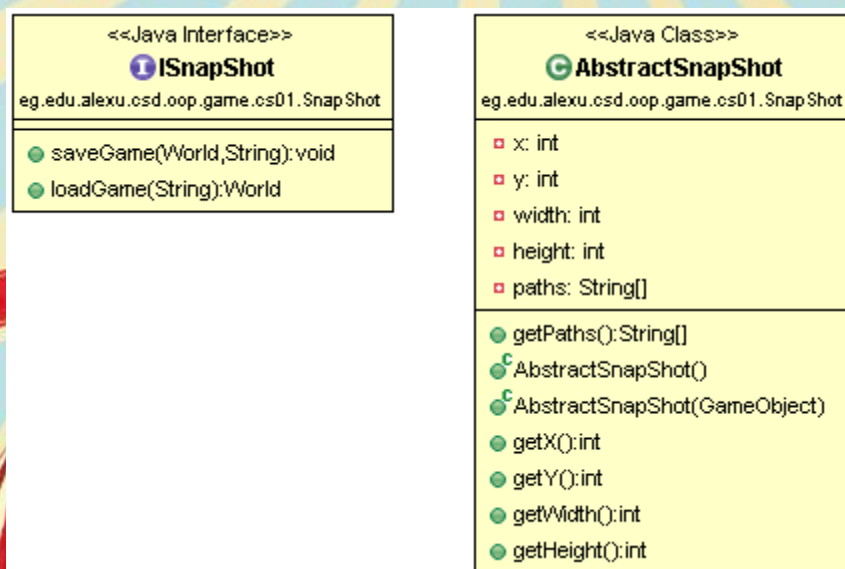
- getRemainingTime(): double
- ^COurGame()
- ^COurGame(GameDifficulty, GameMode)
- getConstantObjects(): List<GameObject>
- getMovableObjects(): List<GameObject>
- getControlableObjects(): List<GameObject>
- getWidth(): int
- getHeight(): int
- getStatus(): String
- getLives(): int
- getScore(): int
- getDifficulty(): GameDifficulty
- getSpeed(): int
- getControlSpeed(): int
- getMode(): GameMode
- setMode(GameMode): void
- getState(): CurrentState
- setState(CurrentState): void
- getDuration(): Duration
- refresh(): boolean
- getSnapShot(): GameSnapShot
- getCounter(): int
- loadGame(GameSnapShot): void

World of plates

Refresh IRrefresh



AbstractSnapShot ISnapShot

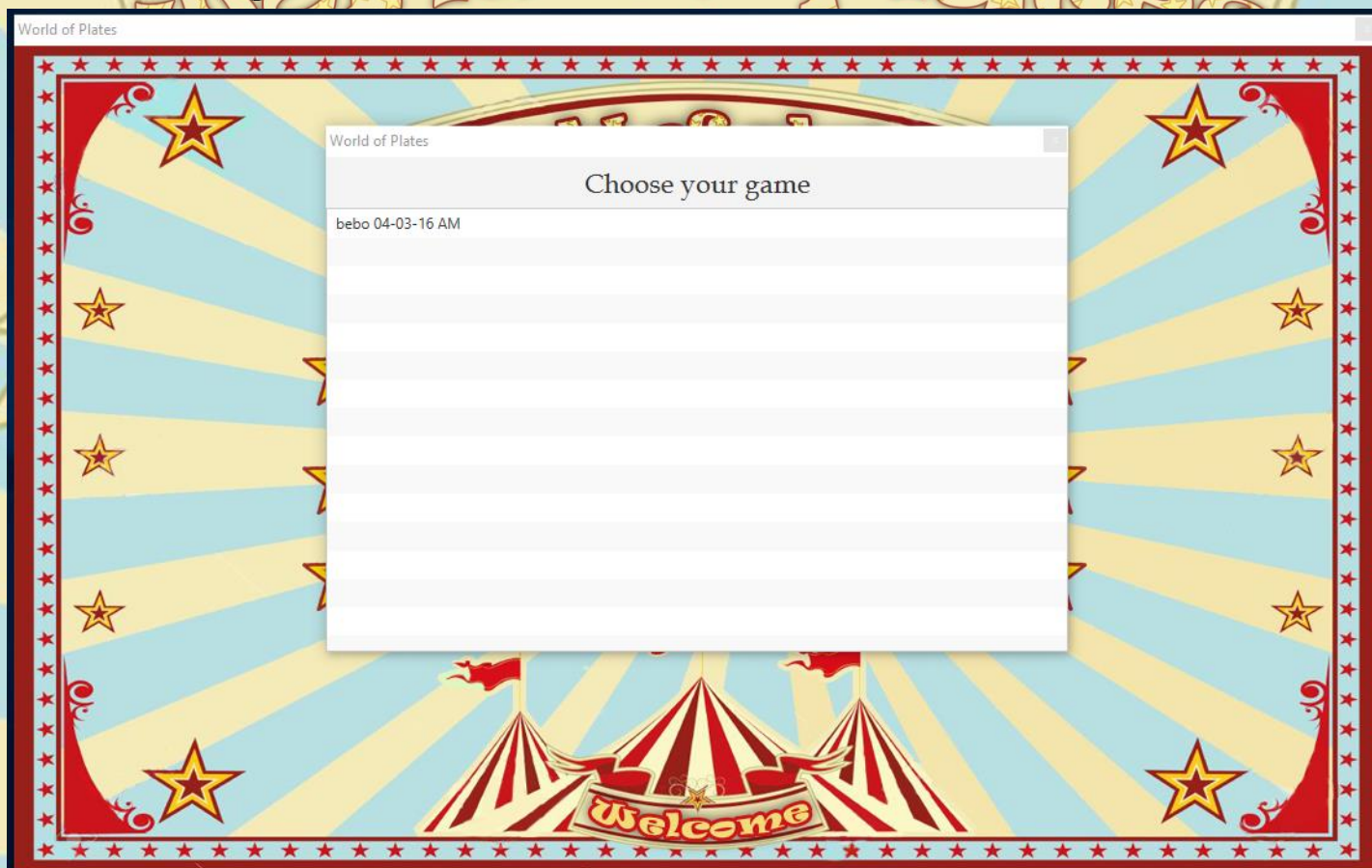


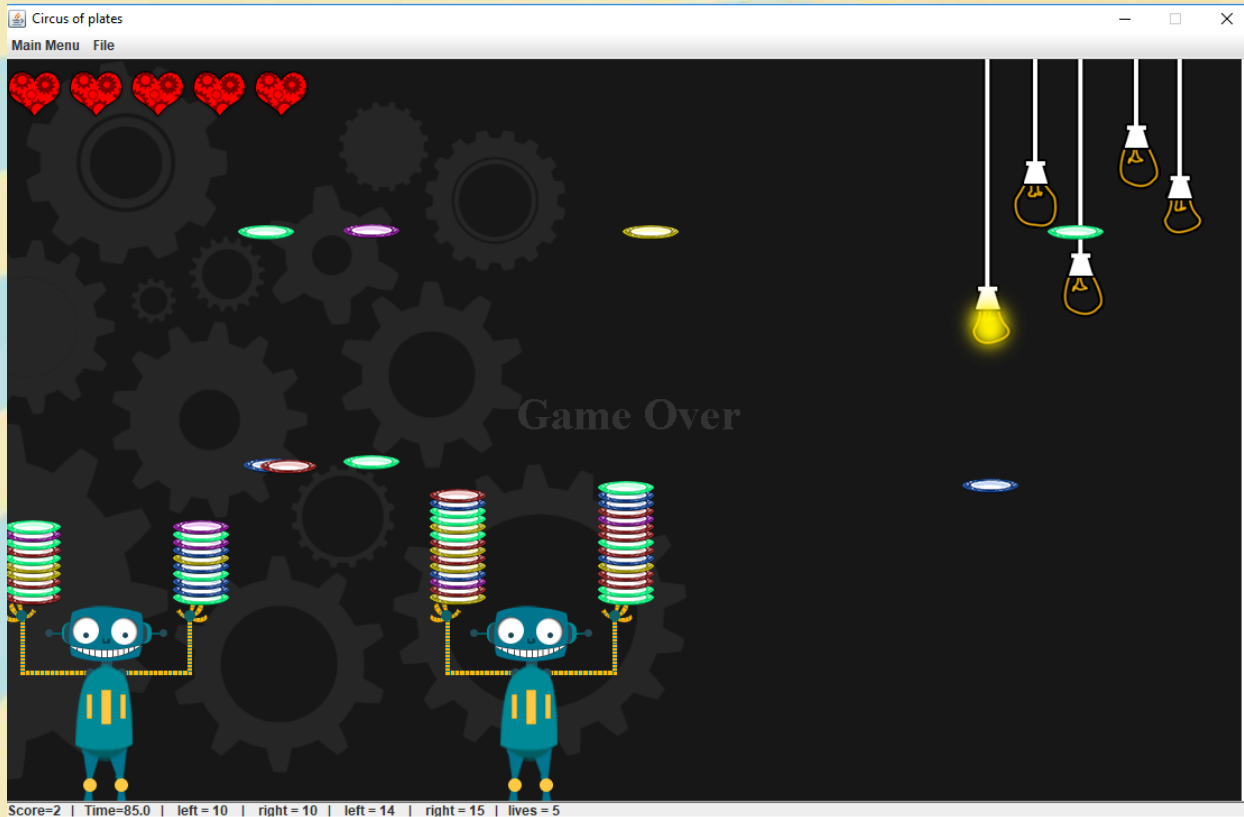
User Manual:

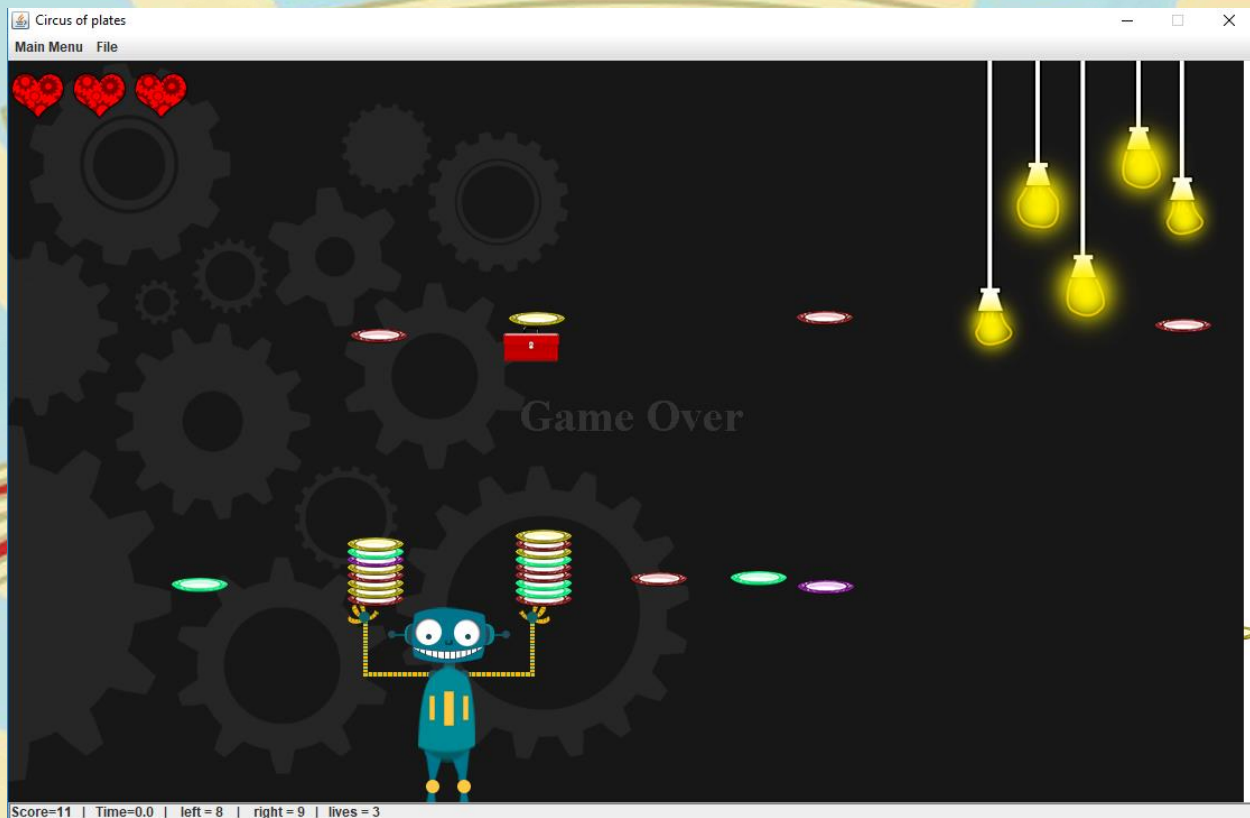
- Choose the mode of the game if it is *easy*, *medium*, or *hard*.
- Choose the mode if it is *Christmas* or *Robot*.
- *You can control your character using arrows.*
- The game can be paused while playing and resuming the game again.
- There are five lives.
- If you hit a bomb your lives decrease by one.
- If you take a present your score increases by one.
- You can choose *new game* after game is over.
- Game is over when one of the stacks of plates of your character is full with 15 plate, when the time ends or when you lose your five lives.

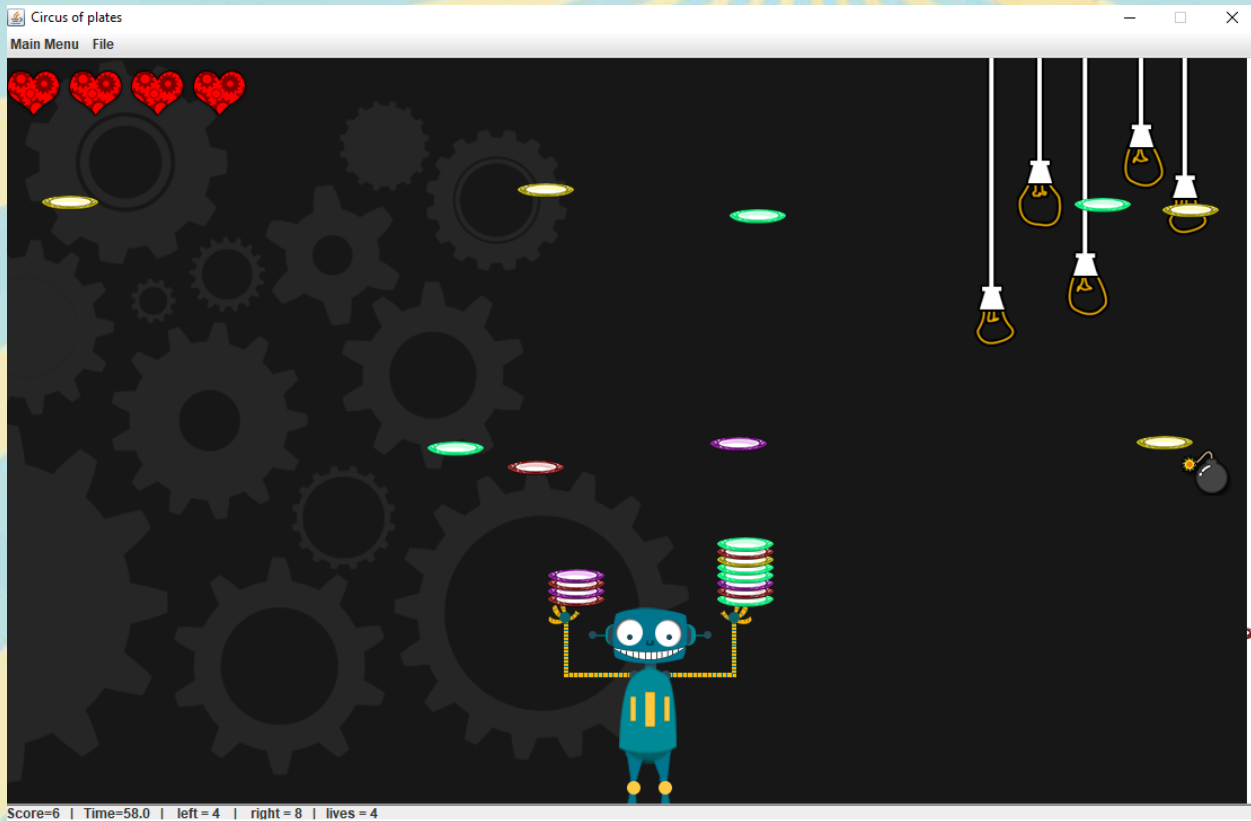
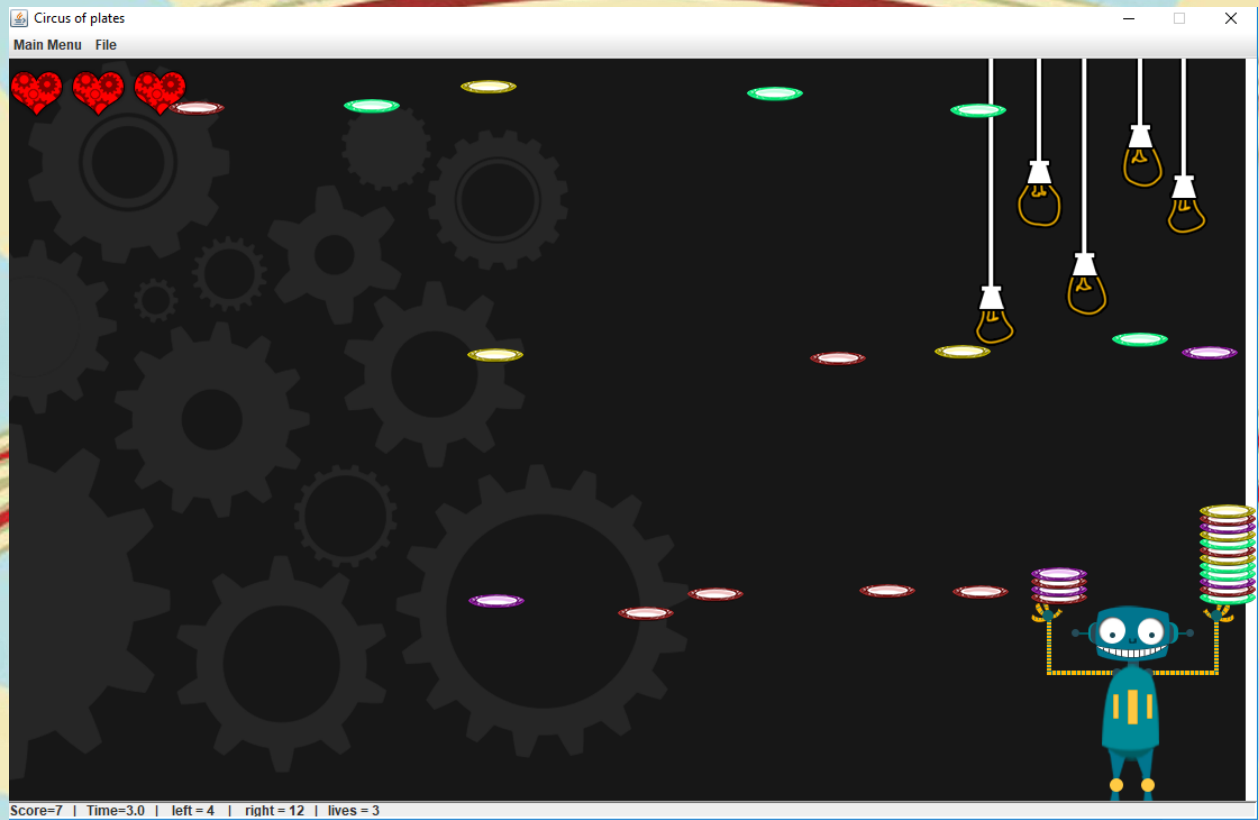


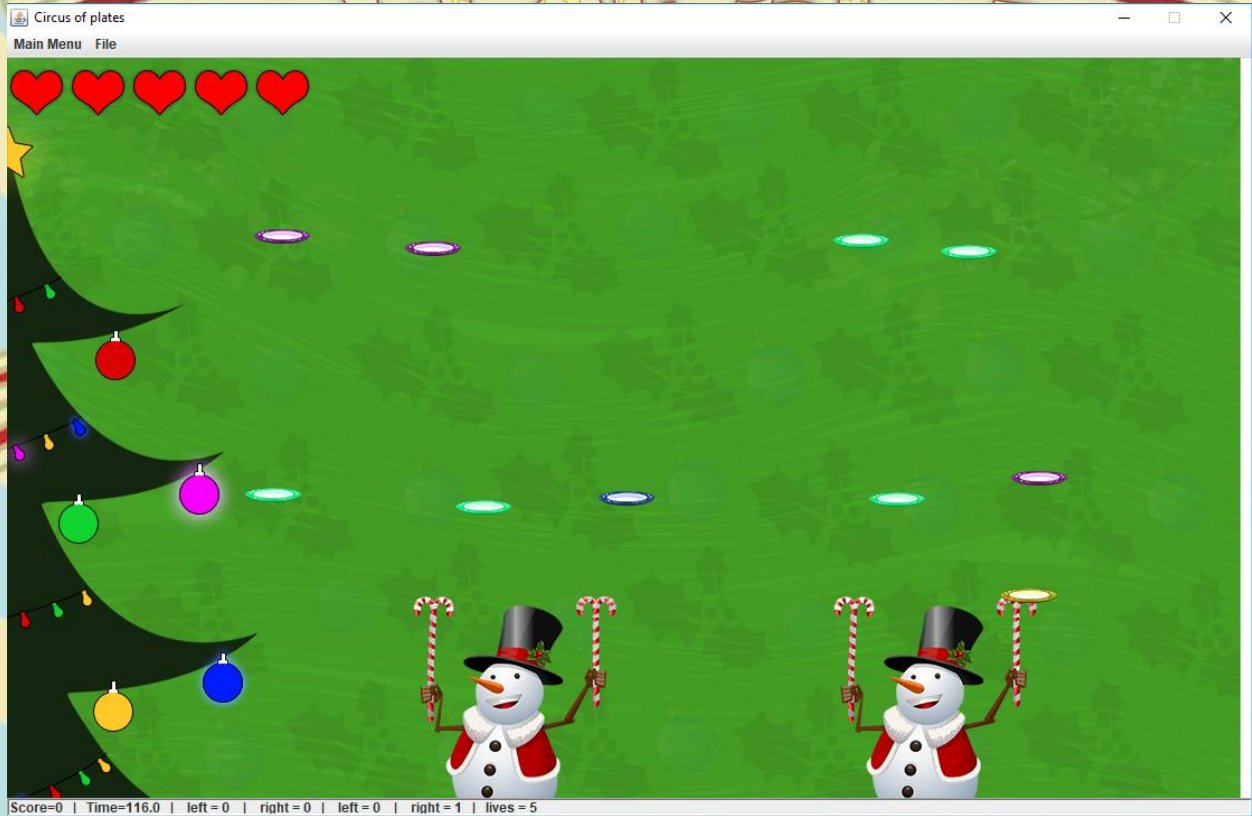
Sample Runs:











Design's Class Diagram:

Sequence Diagram

eg.edu.alien.csd.org.game.cdb1.OurWorld.OurGame.refresh()

