Maze Runner Game

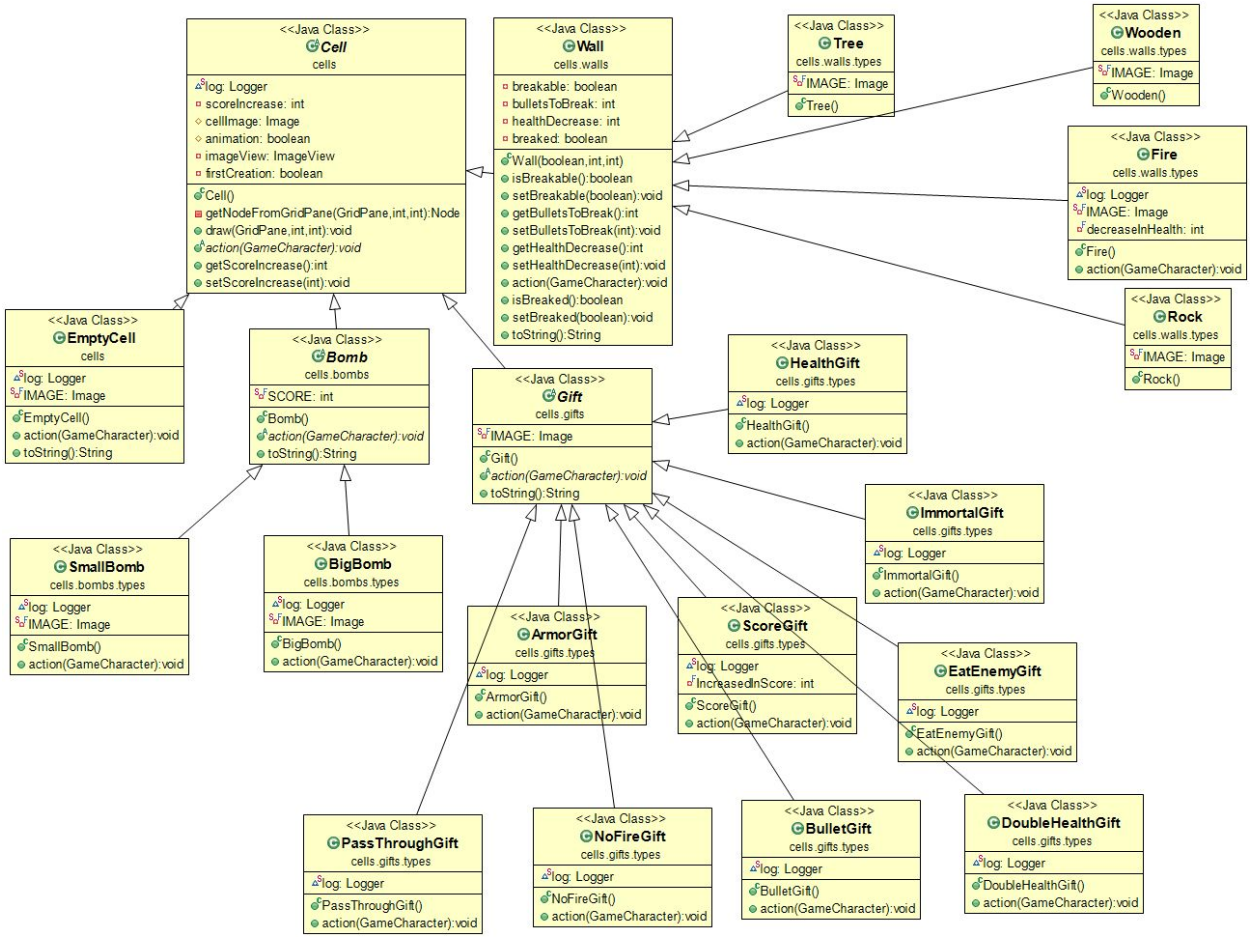
Members :

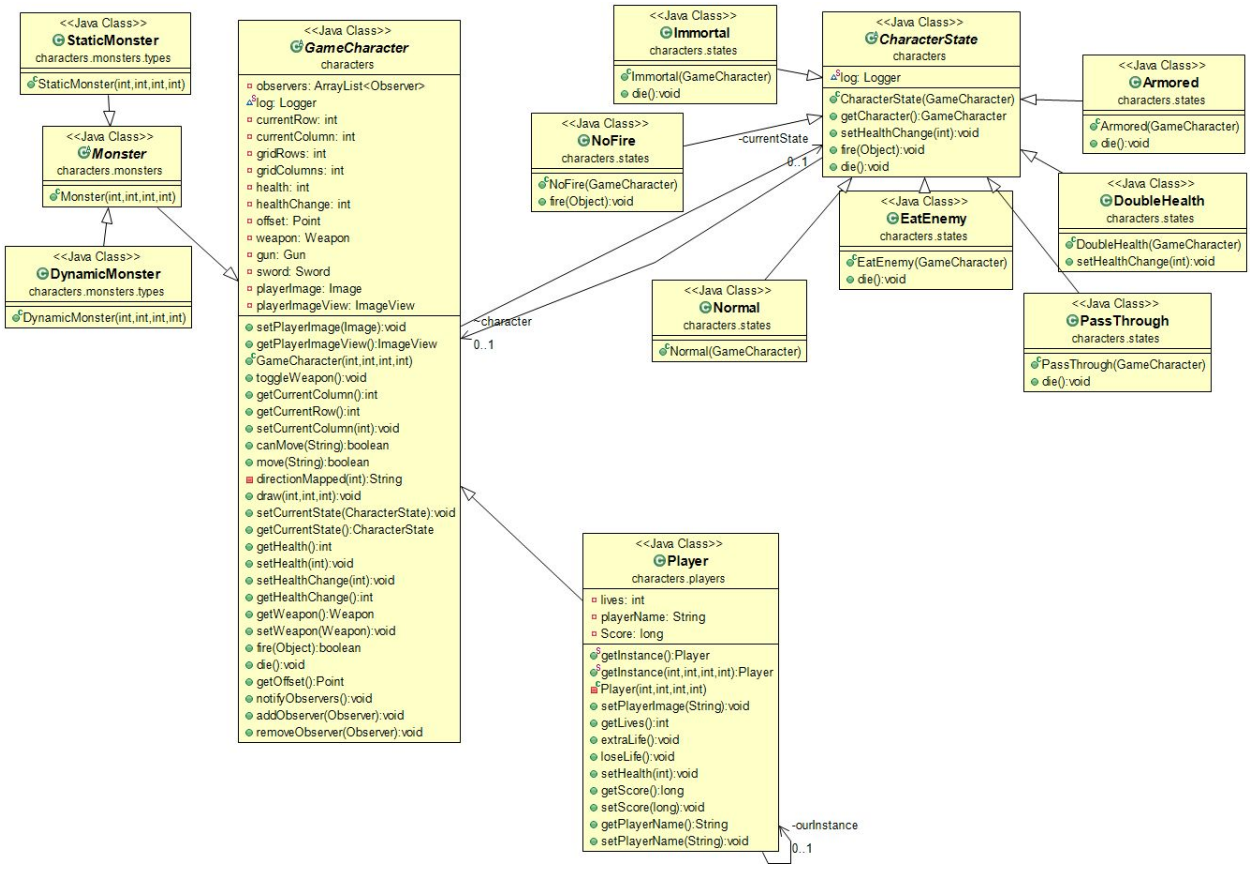
- Mohamed Mashaal Mohamed 60
- Yousef Ali Hassan 73
- Mohamed elsayed sharaf 54
- Mohamed el maghraby mohamed 55

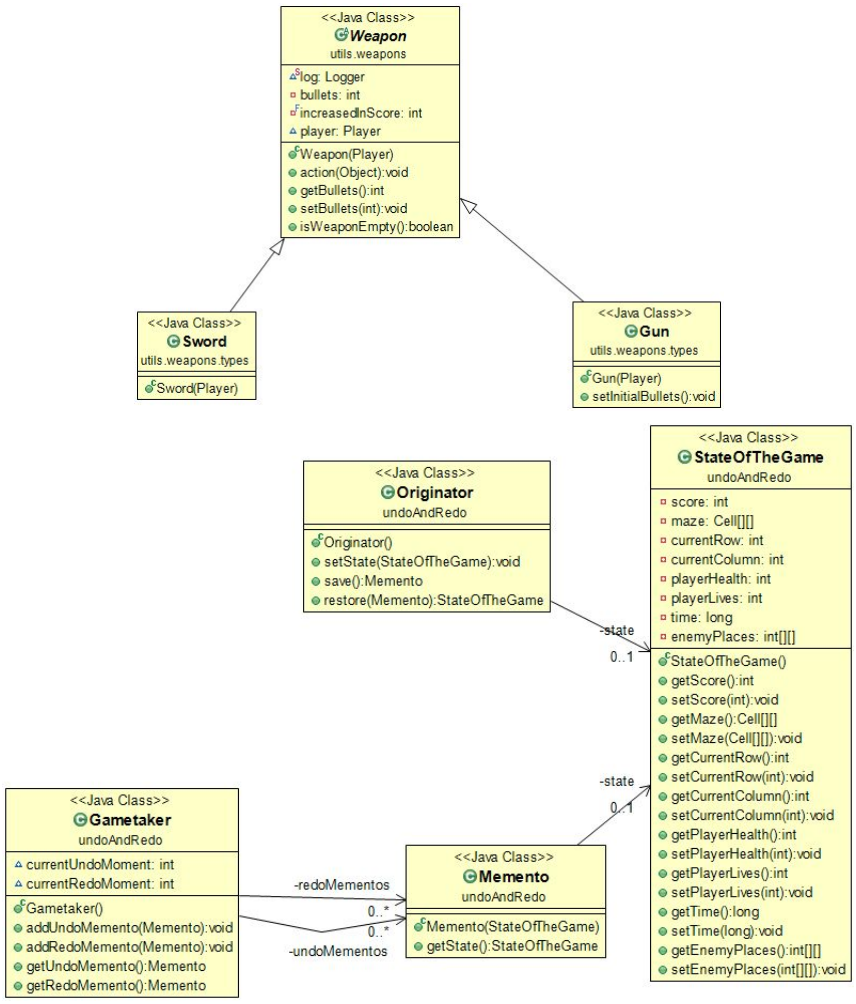
Design description :

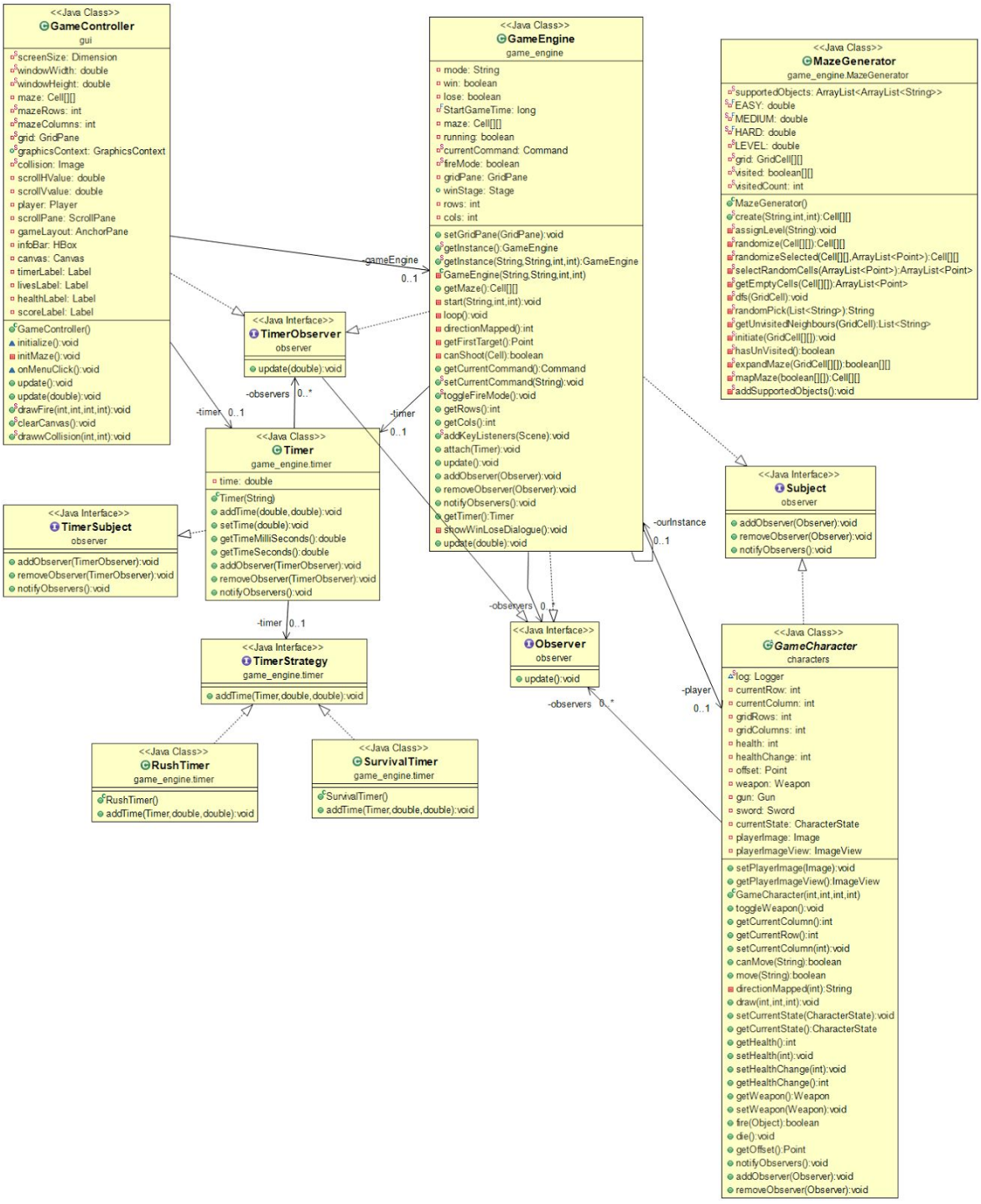
- We used MVC model the model contains of :
    - In the design we assume that each element is a cell except the player and the monsters EX : (wall cell, bomb cell , gift cell. ...etc).
    - Each of these types has sub types EX: (health gift, Armor gift, bullets gift , ...etc).
    - Each of these cell classes has an action function that can be applied on the player when he reach the cell and a draw function to be used in view.
- The controller is game engine class that manage all the game using a game loop, include timer , compute score and manage all actions of the cells.
- The view is done with Javafx that calls the draw functions in each cell and handle uploading images, choosing characters for the player or determining level of the game.

Class diagram :

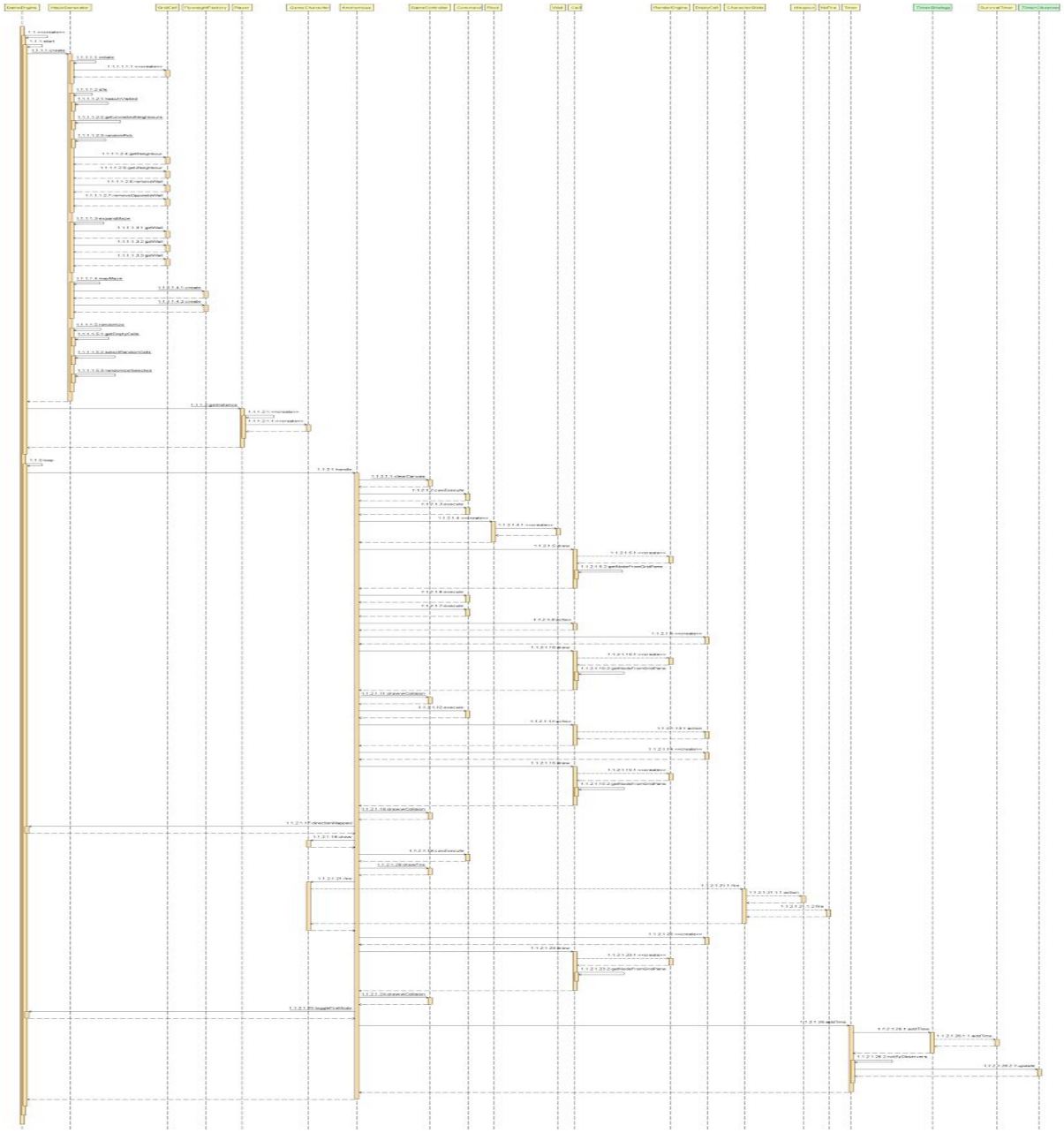



<<Java Class>>
StaticMonster
characters.monsters.types
StaticMonster(int,int,int,int)
<<Java Class>>
Monster
characters.monsters
Monster(int,int,int,int)
<<Java Class>>
DynamicMonster
characters.monsters.types
DynamicMonster(int,int,int,int)
<<Java Class>>
GameCharacter
characters
observers: ArrayList<Observer>
log: Logger
currentRow: int
currentColumn: int
gridRows: int
gridColumns: int
health: int
healthChange: int
offset: Point
weapon: Weapon
gun: Gun
sword: Sword
playerImage: Image
playerImageView: ImageView
setPlayerImage(Image):void
getPlayerImageView():ImageView
GameCharacter(int,int,int,int)
toggleWeapon():void
getCurrentColumn():int
getCurrentRow():int
setCurrentColumn(int):void
canMove(String):boolean
move(String):boolean
directionMapped(int):String
draw(int,int,int):void
setCurrentState(CharacterState):void
getCurrentState():CharacterState
getHealth():int
setHealth(int):void
setHealthChange(int):void
getHealthChange():int
getWeapon():Weapon
setWeapon(Weapon):void
fire(Object):boolean
die():void
getOffset():Point
notifyObservers():void
addObserver(Observer):void
removeObserver(Observer):void
<<Java Class>>
Immortal
characters.states
Immortal(GameCharacter)
die():void
<<Java Class>>
NoFire
characters.states
NoFire(GameCharacter)
fire(Object):void
<<Java Class>>
CharacterState
characters
log: Logger
CharacterState(GameCharacter)
getCharacter():GameCharacter
setHealthChange(int):void
fire(Object):void
die():void
<<Java Class>>
Armored
characters.states
Armored(GameCharacter)
die():void
<<Java Class>>
EatEnemy
characters.states
EatEnemy(GameCharacter)
die():void
<<Java Class>>
DoubleHealth
characters.states
DoubleHealth(GameCharacter)
setHealthChange(int):void
<<Java Class>>
Normal
characters.states
Normal(GameCharacter)
<<Java Class>>
PassThrough
characters.states
PassThrough(GameCharacter)
die():void
-currentState
0..1
-character
0..1
<<Java Class>>
Player
characters.players
lives: int
playerName: String
Score: long
getInstance():Player
getInstance(int,int,int,int):Player
Player(int,int,int,int)
setPlayerImage(String):void
getLives():int
extraLife():void
loseLife():void
setHealth(int):void
getScore():long
setScore(long):void
getPlayerName():String
setPlayerName(String):void
-ourInstance
0..1


<<Java Class>>
Weapon
utils.weapons
log: Logger
bullets: int
increasedInScore: int
player: Player
Weapon(Player)
action(Object):void
getBullets():int
setBullets(int):void
isWeaponEmpty():boolean
<<Java Class>>
Sword
utils.weapons.types
Sword(Player)
<<Java Class>>
Gun
utils.weapons.types
Gun(Player)
setInitialBullets():void
<<Java Class>>
Originator
undoAndRedo
Originator()
setState(StateOfTheGame):void
save():Memento
restore(Memento):StateOfTheGame
<<Java Class>>
StateOfTheGame
undoAndRedo
score: int
maze: Cell[][]
currentRow: int
currentColumn: int
playerHealth: int
playerLives: int
time: long
enemyPlaces: int[][]
StateOfTheGame()
getScore():int
setScore(int):void
getMaze():Cell[][]
setMaze(Cell[][]):void
getCurrentRow():int
setCurrentRow(int):void
getCurrentColumn():int
setCurrentColumn(int):void
getPlayerHealth():int
setPlayerHealth(int):void
getPlayerLives():int
setPlayerLives(int):void
getTime():long
setTime(long):void
getEnemyPlaces():int[][]
setEnemyPlaces(int[][]):void
-state
0..1
-state
0..1
<<Java Class>>
Gametaker
undoAndRedo
currentUndoMoment: int
currentRedoMoment: int
Gametaker()
addUndoMemento(Memento):void
addRedoMemento(Memento):void
getUndoMemento():Memento
getRedoMemento():Memento
-redoMementos
0..*
0..*
-undoMementos
<<Java Class>>
Memento
undoAndRedo
Memento(StateOfTheGame)
getState():StateOfTheGame


<<Java Class>>
GameController
gui
screenSize: Dimension
windowWidth: double
windowHeight: double
maze: Cell[][]
mazeRows: int
mazeColumns: int
grid: GridPane
graphicsContext: GraphicsContext
collision: Image
scrollHValue: double
scrollVValue: double
player: Player
scrollPane: ScrollPane
gameLayout: AnchorPane
infoBar: HBox
canvas: Canvas
timerLabel: Label
livesLabel: Label
healthLabel: Label
scoreLabel: Label
GameController()
initialize():void
initMaze():void
onMenuClick():void
update():void
update(double):void
drawFire(int,int,int,int):void
clearCanvas():void
drawwCollision(int,int):void
<<Java Class>>
GameEngine
game_engine
mode: String
win: boolean
lose: boolean
StartGameTime: long
maze: Cell[][]
running: boolean
currentCommand: Command
fireMode: boolean
gridPane: GridPane
winStage: Stage
rows: int
cols: int
setGridPane(GridPane):void
getInstance():GameEngine
getInstance(String,String,int,int):GameEngine
GameEngine(String,String,int,int)
getMaze():Cell[][]
start(String,int,int):void
loop():void
directionMapped():int
getFirstTarget():Point
canShoot(Cell):boolean
getCurrentCommand():Command
setCurrentCommand(String):void
toggleFireMode():void
getRows():int
getCols():int
addKeyListeners(Scene):void
attach(Timer):void
update():void
addObserver(Observer):void
removeObserver(Observer):void
notifyObservers():void
getTimer():Timer
showWinLoseDialogue():void
update(double):void
<<Java Class>>
MazeGenerator
game_engine.MazeGenerator
supportedObjects: ArrayList<ArrayList<String>>
EASY: double
MEDIUM: double
HARD: double
LEVEL: double
grid: GridCell[][]
visited: boolean[][]
visitedCount: int
MazeGenerator()
create(String,int,int):Cell[][]
assignLevel(String):void
randomize(Cell[][]):Cell[][]
randomizeSelected(Cell[][],ArrayList<Point>):Cell[][]
selectRandomCells(ArrayList<Point>):ArrayList<Point>
getEmptyCells(Cell[][]):ArrayList<Point>
dfs(GridCell):void
randomPick(List<String>):String
getUnvisitedNeighbours(GridCell):List<String>
initiate(GridCell[][]):void
hasUnVisited():boolean
expandMaze(GridCell[][]):boolean[][]
mapMaze(boolean[][]):Cell[][]
addSupportedObjects():void
<<Java Interface>>
TimerObserver
observer
update(double):void
-observers 0..*
-timer 0..1
-timer 0..1
<<Java Class>>
Timer
game_engine.timer
time: double
Timer(String)
addTime(double,double):void
setTime(double):void
getTimeMilliSeconds():double
getTimeSeconds():double
addObserver(TimerObserver):void
removeObserver(TimerObserver):void
notifyObservers():void
<<Java Interface>>
TimerSubject
observer
addObserver(TimerObserver):void
removeObserver(TimerObserver):void
notifyObservers():void
-timer 0..1
<<Java Interface>>
TimerStrategy
game_engine.timer
addTime(Timer,double,double):void
<<Java Class>>
RushTimer
game_engine.timer
RushTimer()
addTime(Timer,double,double):void
<<Java Class>>
SurvivalTimer
game_engine.timer
SurvivalTimer()
addTime(Timer,double,double):void
-gameEngine
0..1
-ourInstance
0..1
<<Java Interface>>
Subject
observer
addObserver(Observer):void
removeObserver(Observer):void
notifyObservers():void
-observers 0..*
<<Java Interface>>
Observer
observer
update():void
-observers 0..*
-player
0..1
<<Java Class>>
GameCharacter
characters
log: Logger
currentRow: int
currentColumn: int
gridRows: int
gridColumns: int
health: int
healthChange: int
offset: Point
weapon: Weapon
gun: Gun
sword: Sword
currentState: CharacterState
playerImage: Image
playerImageView: ImageView
setPlayerImage(Image):void
getPlayerImageView():ImageView
GameCharacter(int,int,int,int)
toggleWeapon():void
getCurrentColumn():int
getCurrentRow():int
setCurrentColumn(int):void
canMove(String):boolean
move(String):boolean
directionMapped(int):String
draw(int,int,int):void
setCurrentState(CharacterState):void
getCurrentState():CharacterState
getHealth():int
setHealth(int):void
setHealthChange(int):void
getHealthChange():int
getWeapon():Weapon
setWeapon(Weapon):void
fire(Object):boolean
die():void
getOffset():Point
notifyObservers():void
addObserver(Observer):void
removeObserver(Observer):void

Sequence diagram :



Design patterns used:

1. ## Factory design pattern:

   Usd in creating cells, players, monsters, bombs, gifts , etc.

2. ## Flyweight design pattern:

   a. It manage the factory as we create only one object from each cell type and use it with multiple references.

   b. Used for images creation - just use on image object for each object in the game -.

3. ## Singleton design pattern :

   Used in creating a single player and in the game engine as there must be one game engine for the game.

4. ## Command design pattern :

   Used for moving the player, scroll the image and fire weapon.

5. ## Observer design pattern :

   - The game engine observe the player.
   - Game controller observe the timer.
   - Game controller observe the game engine.
   - The player observe the weapon.

6. **State design pattern :**

   Define states for the player as the functions differ depending on the state of the player, EX : if the player has armor gift he can't die.

7. **Strategy design pattern :**

   For the monsters movement as each monster has its way of tracing the player or shooting it.

8. **Memento design pattern :**

   For undo and redo by saving the state of the game after each action in an arraylist.

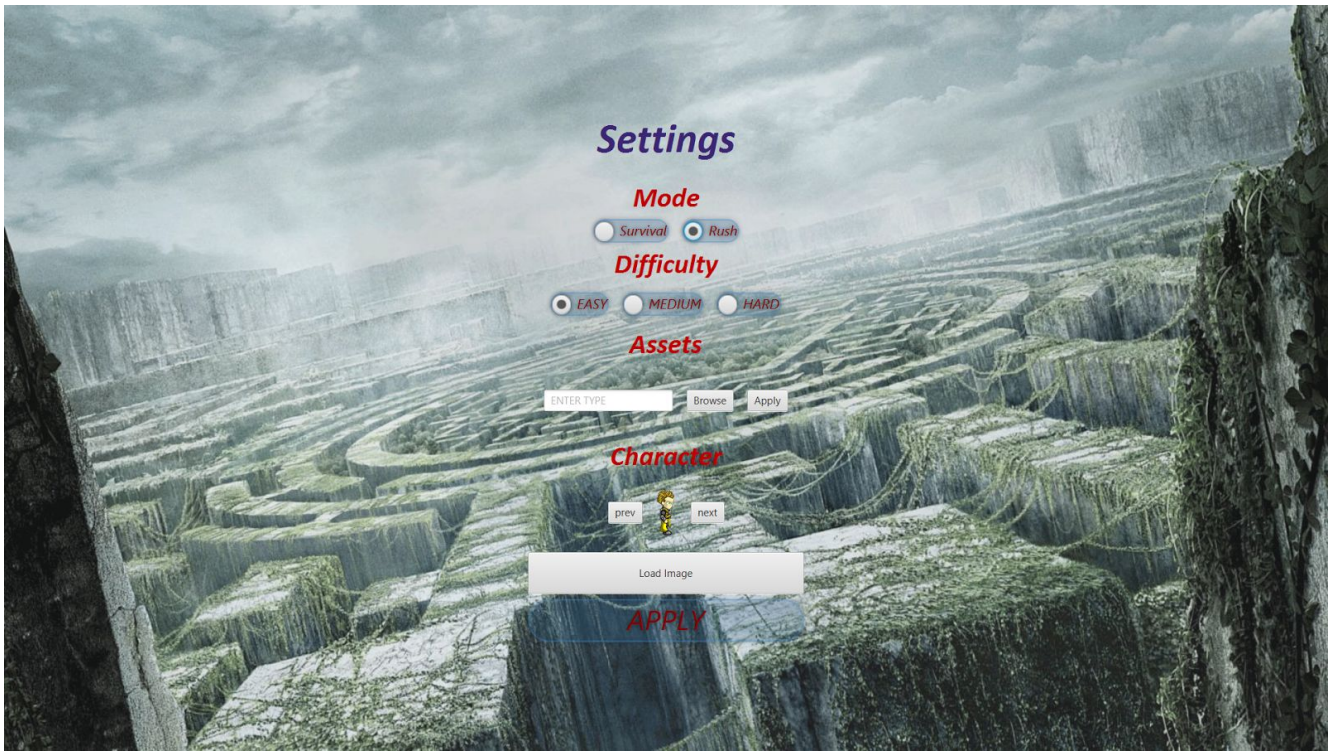
9. **Iterator design pattern :**

   Used to iterate on arraylists elements.

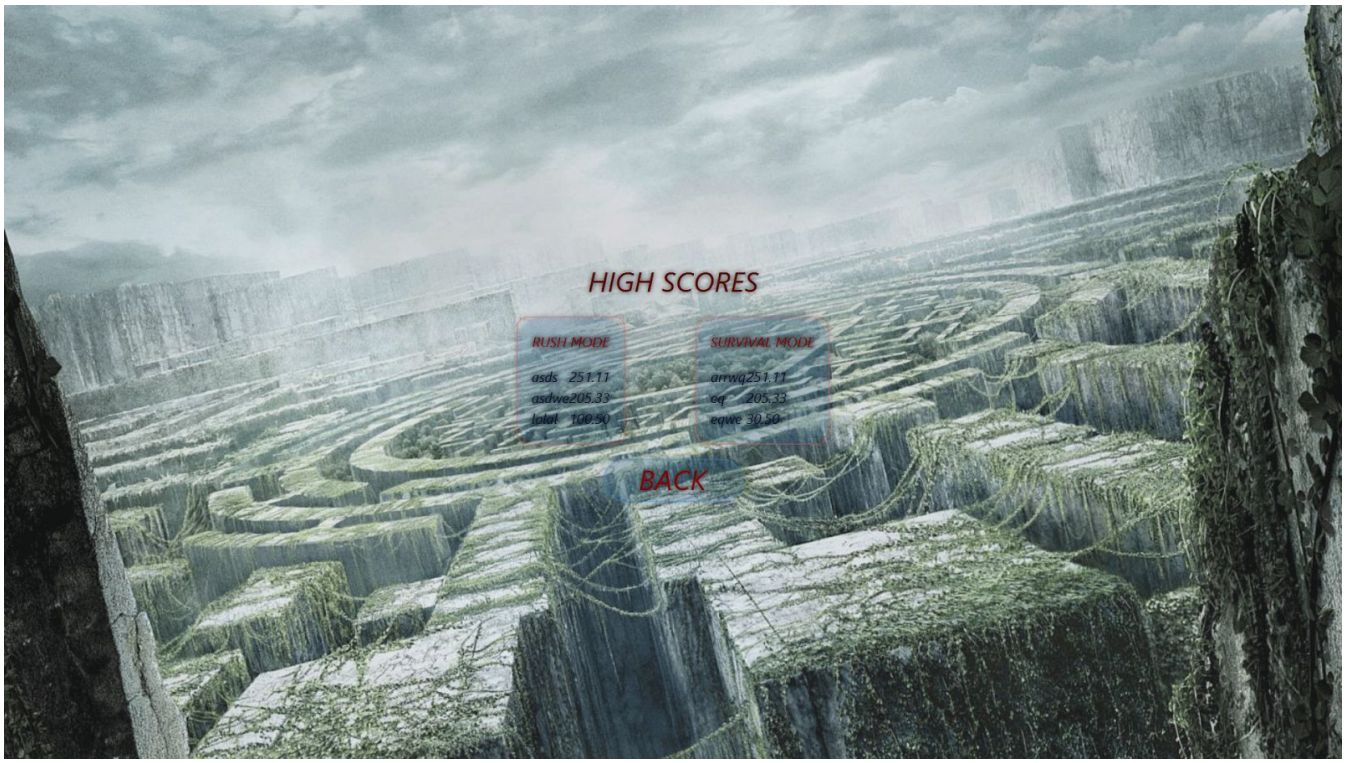
Snapshots of the GUI:

Main :

Settings :


Settings
Mode
Survival
Rush
Difficulty
EASY
MEDIUM
HARD
Assets
ENTER TYPE
Browse
Apply
Character
prev
next
Load Image
APPLY

Score Board :


HIGH SCORES
RUSH MODE
asds 251.11
asdwe205.33
lolol 100.50
SURVIVAL MODE
arrwq251.11
eq 205.33
eqwe 30.50
BACK

Game view :


MENU
00:09
Lives : 3
Health : 100
Score : 0
State : Normal
Weapon : Sword
Bullets : 0

User guide :

1. Run the game from the jar.
2. From settings select :
    a. Level of difficulty.
    b. Player avatar.
    c. Upload images if you want to use instead of default ones.
3. Click start the game to play.
4. Move by using (W : up , D : right , A : left , S : down).
5. Fire using ‘X’ and determining the direction with (W,A,S,D).
6. Toggle weapon (between sword and gun) using ‘T’.
7. Scroll the page by mouse or the buttons on the top left of the screen.
8. You can show the high score from the main menu.