Get Out

(if you can :D)

Aya Ashraf Saber Mohamed (02)
Sarah Ahmed Eldafrawy (29)
AbdelRahman Ibrahim soliman (35)
Mahmoud Gamal (61)
Mohamed Kamal (59)

Problem statement:

It is required to implement a maze runner game with various levels and difficulties. Designing this game must be completed first using design patterns to make it more structured and easy to implement and test. The game should contain different levels, gifts, bombs and walls.

User Guide:

The user run the application and then the Main Menu is displayed. A window is displayed allowing user to pick the hero, and starting level. Afterwards, the game starts and the hero is by default in position (1,1). He can moves using keyboard arrows and can shoot using space, also pick different weapons collected using next and previous keys (Q, E). The user must watch out for moving monsters, he may shoot them. There is also different kind of bombs that varies in damage. Different objects of collectors are distributed around the maze randomly such as gifts, shields and weapons. To win the game, the hero must find the key and collect it then opens the gate at the bottom right of the maze.

Allowed Movements are using keyboard keys:

- key UP: moves hero step up
- key DOWN: moves hero step down
- key LEFT: moves hero step left
- key SPACE: shoots bullets
- key E: holds next weapon
- key Q: hold previous weapon
- key V: zoom in and out

Overview :

Modularity is the main point of the design every logic has its own module:

The MVC model is used as following:

The model ”M” part:

It is partitioned into the following stages:

- Deciding the needed level.
- Loading the level properties.
- Generating the maze structure.

- Generating the maze components according to the required level.
- Assembling the components in the maze.
- Creating a façade object and send it to the controller
    - The façade object handles the changes that occurs in the maze “motion of hero and monsters, destruction of walls and picking gifts”.
- The maze components are all drawables that can be drawn on a canvas.
- They are divided into 3 parts:
    - Moving objects: heroes and monsters.
    - Pickables: gifts, weapons and shields.
    - Obstacles: walls, bombs and traps.
- Each part is implemented in his own package and treats others using interfaces.
- Monsters in levels have different motion strategies.

The controller “C” part:

It is partitioned into various controllers with defined roles for each one, for example the canvas controller that organizes the drawables on the canvases we have “4 canvases”.

The menu controller which organizes the interface to the user and initiates the call for

The model part.

The view ”V” part :

It consists of various scenes along with their fxml files to display view components. The main view layout is the game window where the player can move the hero using keyboard arrows, shooting using space and performing other actions too. The view consists of multiple layers (canvases) and using observers we can update only a part of the layer required to change. Another layout is also displayed for the user at the beginning to allow the user to choose his hero.

Design patterns used:

We used 14 design patterns and there is room for another 2:

- Observer DP
    - We implemented the observer pattern 8 times:
        - Win and lose observer.
        - Bomb explosion observer.
        - Hero state observer.
        - Maze layers observer.
        - Bullets motion observer.
        - Monster motion observer. “view”
        - Moving object observer. ”back end maze”
        - Death observer.

- Strategy DP:
  - Monsters shooting strategies.

- State DP:
  - Used in the direction states for drawing images and for shooting.
  - Four states one for each direction.

- Prototype DP:
  - Bullets are not created on taking ammo or carrying a new weapon it is created when the user decides to shoot and since we have many and various types of weapons with different bullet capacities “some are 6 others are 30” and so on so object pool is not the optimal choice so a clonable bullet was our choice.

- Flyweight DP:
  - Used in loading sprite sheets by loading the image only once and then adding frames in an ArrayList to be called later, so images are only loaded once when called for the first time.

- Bridge DP:
  - The bridge design pattern allows what it is like a Cartesian product between two types where any concrete handler can hold any concrete pickable and any concrete pickable can be handled by and concrete handler.

- Builder DP:
  - This design pattern concerns mostly the building process of a maze for a given level properties.

- Façade DP:
  - All the implementation and interaction of objects are all hidden from the view using a façade object called Game loop.

- Factory DP:
  - It is used in the dynamic loading of level properties.

- Dynamic Linkage DP:
  - Load and instantiate class automatic for supporting extensions

- Singleton DP:
  - The direction states are singleton as well as the game core.

- Command DP:
  - Used in motion commands given to any moving object “hero or monsters”

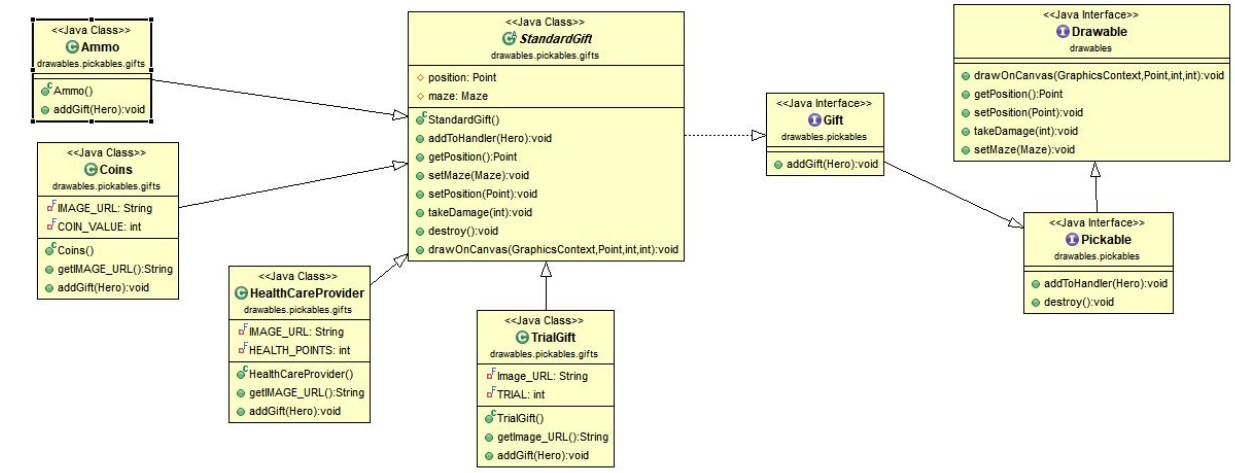
- Iterator DP:
  - Collection of weapons the hero has is iterated on by the use of java iterator.

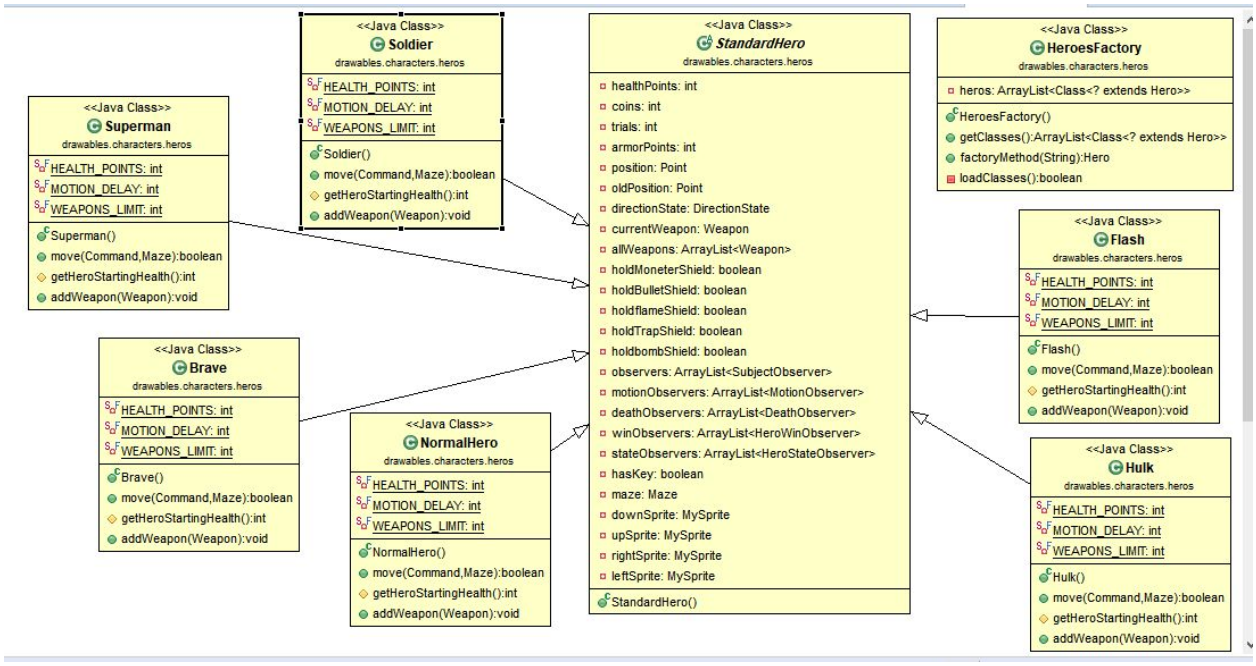
- NULL object DP:
  - Instead of using null as a value, dummy roads are found in the maze structure. It takes no damage and do no harm.

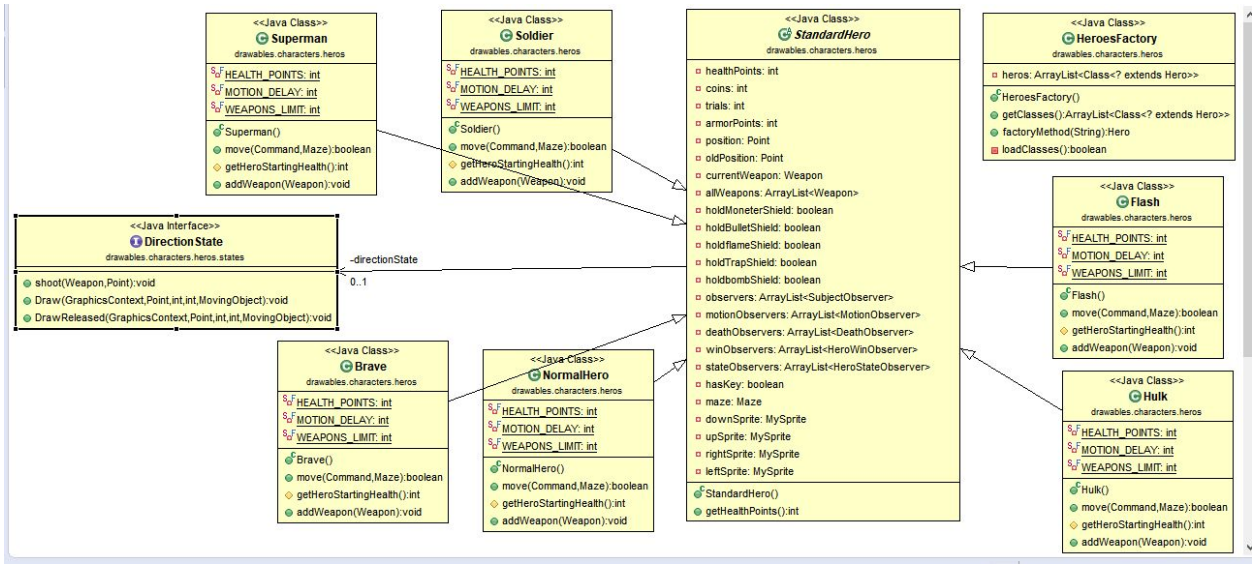
- The following design patterns are not implemented but can be added :
  - Decorator DP:
    - The making of special levels that decorates the normal levels with gifts and weapons
  - Memento DP:
    - For taking in game snapshots.
    - The code is ready but not integrated with the game.

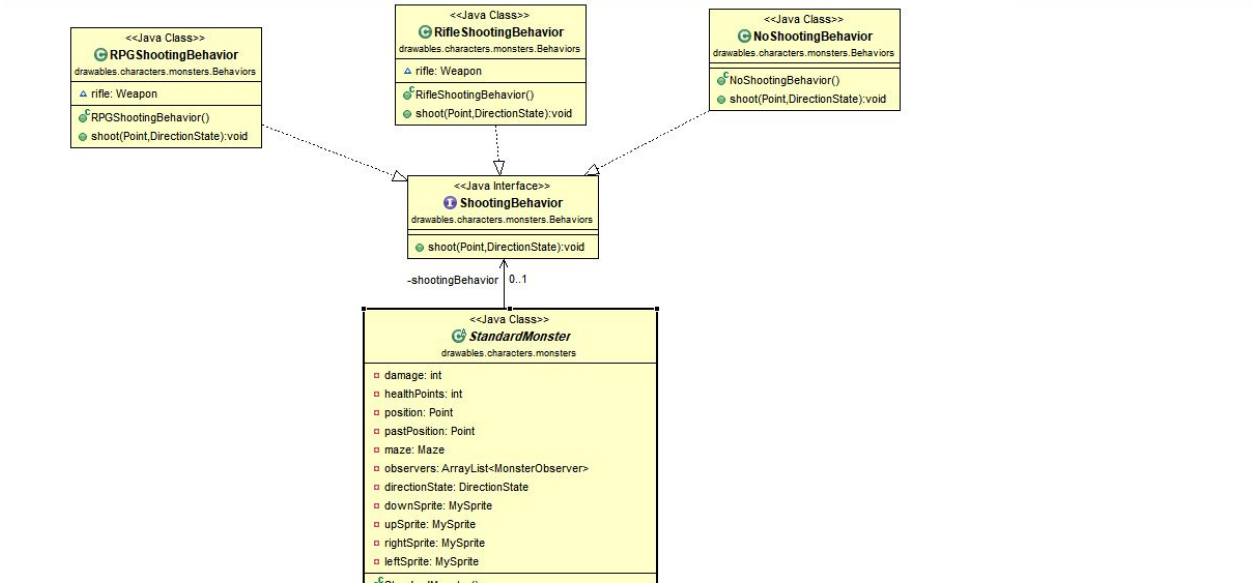
UML Diagrams:

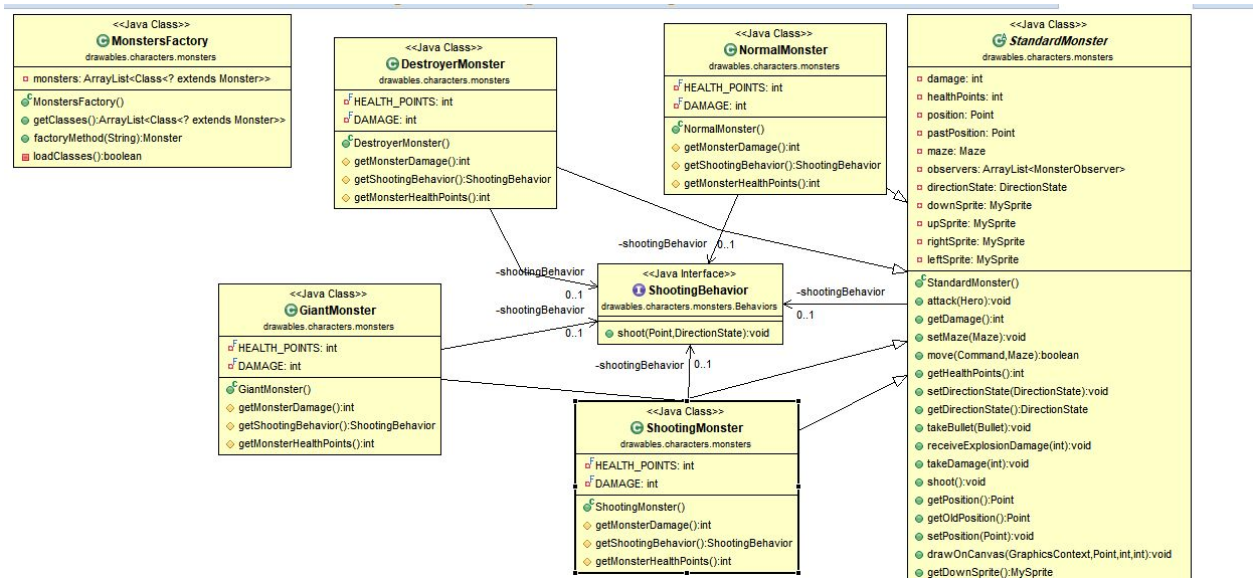
1. **Class Diagram :**
   a. **Drawables:**

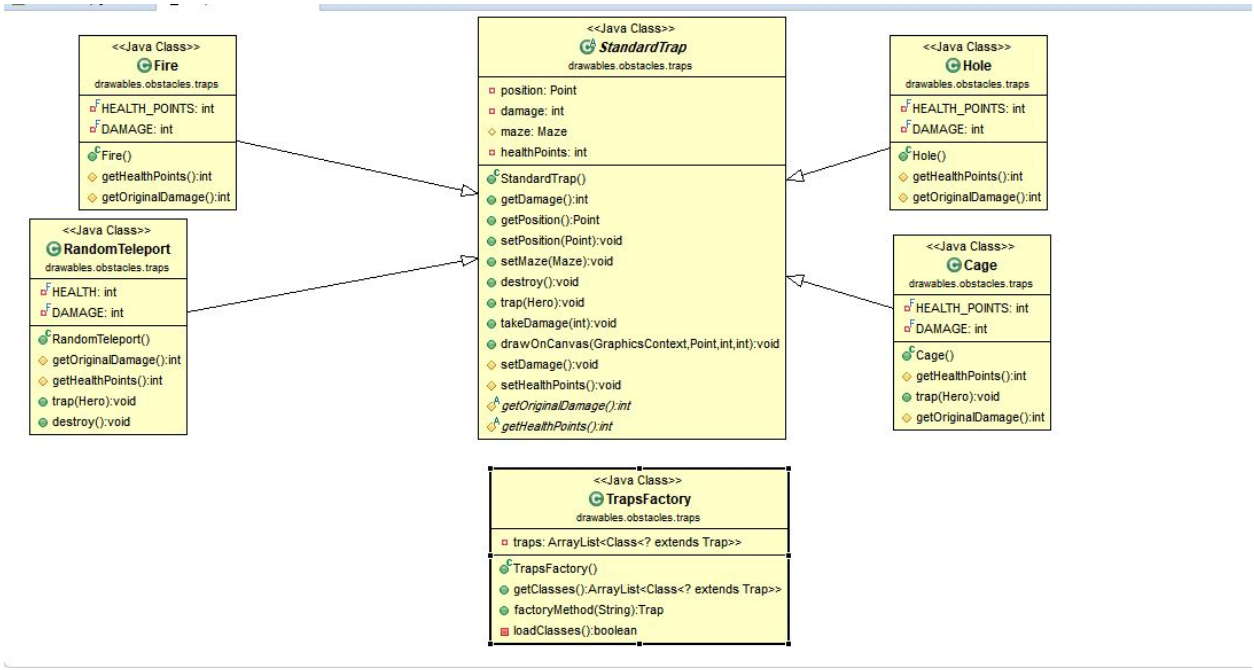

<<Java Class>>
Ammo
drawables.pickables.gifts
Ammo()
addGift(Hero):void
<<Java Class>>
Coins
drawables.pickables.gifts
IMAGE_URL: String
COIN_VALUE: int
Coins()
getIMAGE_URL():String
addGift(Hero):void
<<Java Class>>
HealthCareProvider
drawables.pickables.gifts
IMAGE_URL: String
HEALTH_POINTS: int
HealthCareProvider()
getIMAGE_URL():String
addGift(Hero):void
<<Java Class>>
StandardGift
drawables.pickables.gifts
position: Point
maze: Maze
StandardGift()
addToHandler(Hero):void
getPosition():Point
setMaze(Maze):void
setPosition(Point):void
takeDamage(int):void
destroy():void
drawOnCanvas(GraphicsContext,Point,int,int):void
<<Java Class>>
TrialGift
drawables.pickables.gifts
Image_URL: String
TRIAL: int
TrialGift()
getImage_URL():String
addGift(Hero):void
<<Java Interface>>
Gift
drawables.pickables
addGift(Hero):void
<<Java Interface>>
Drawable
drawables
drawOnCanvas(GraphicsContext,Point,int,int):void
getPosition():Point
setPosition(Point):void
takeDamage(int):void
setMaze(Maze):void
<<Java Interface>>
Pickable
drawables.pickables
addToHandler(Hero):void
destroy():void

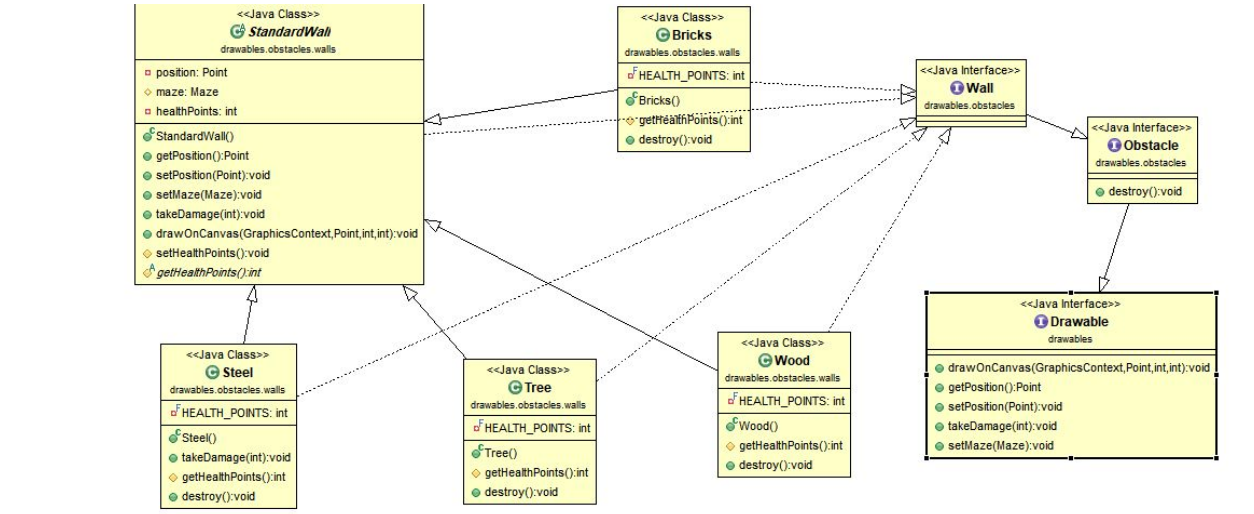

<<Java Class>>
Superman
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Superman()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
Soldier
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Soldier()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
Brave
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Brave()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
NormalHero
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
NormalHero()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
StandardHero
drawables.characters.heros
healthPoints: int
coins: int
trials: int
armorPoints: int
position: Point
oldPosition: Point
directionState: DirectionState
currentWeapon: Weapon
allWeapons: ArrayList<Weapon>
holdMoneterShield: boolean
holdBulletShield: boolean
holdflameShield: boolean
holdTrapShield: boolean
holdbombShield: boolean
observers: ArrayList<SubjectObserver>
motionObservers: ArrayList<MotionObserver>
deathObservers: ArrayList<DeathObserver>
winObservers: ArrayList<HeroWinObserver>
stateObservers: ArrayList<HeroStateObserver>
hasKey: boolean
maze: Maze
downSprite: MySprite
upSprite: MySprite
rightSprite: MySprite
leftSprite: MySprite
StandardHero()
<<Java Class>>
HeroesFactory
drawables.characters.heros
heros: ArrayList<Class<? extends Hero>>
HeroesFactory()
getClasses():ArrayList<Class<? extends Hero>>
factoryMethod(String):Hero
loadClasses():boolean
<<Java Class>>
Flash
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Flash()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
Hulk
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Hulk()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void

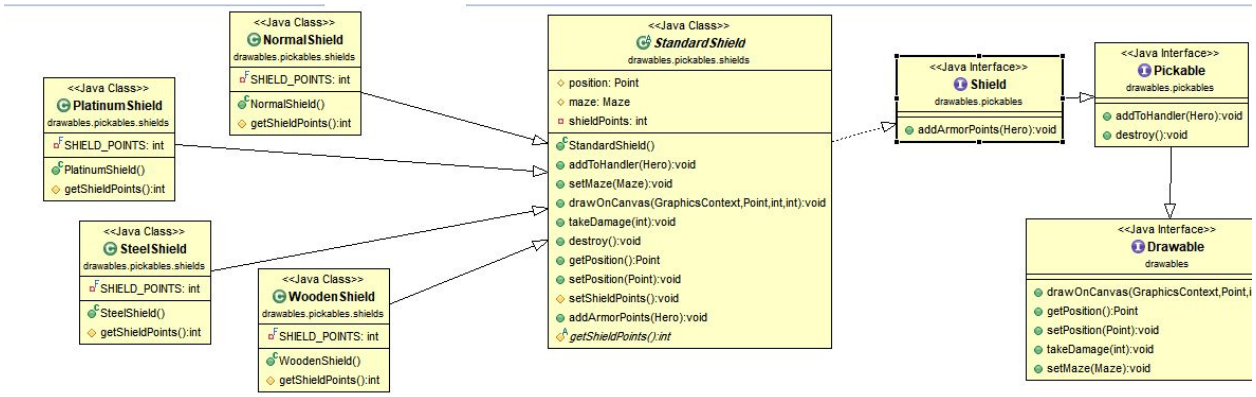

<<Java Class>>
Superman
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Superman()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
Soldier
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Soldier()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
StandardHero
drawables.characters.heros
healthPoints: int
coins: int
trials: int
armorPoints: int
position: Point
oldPosition: Point
currentWeapon: Weapon
allWeapons: ArrayList<Weapon>
holdMoneterShield: boolean
holdBulletShield: boolean
holdflameShield: boolean
holdTrapShield: boolean
holdbombShield: boolean
observers: ArrayList<SubjectObserver>
motionObservers: ArrayList<MotionObserver>
deathObservers: ArrayList<DeathObserver>
winObservers: ArrayList<HeroWinObserver>
stateObservers: ArrayList<HeroStateObserver>
hasKey: boolean
maze: Maze
downSprite: MySprite
upSprite: MySprite
rightSprite: MySprite
leftSprite: MySprite
StandardHero()
getHealthPoints():int
<<Java Class>>
HeroesFactory
drawables.characters.heros
heros: ArrayList<Class<? extends Hero>>
HeroesFactory()
getClasses():ArrayList<Class<? extends Hero>>
factoryMethod(String):Hero
loadClasses():boolean
<<Java Interface>>
DirectionState
drawables.characters.heros.states
shoot(Weapon,Point):void
Draw(GraphicsContext,Point,int,int,MovingObject):void
DrawReleased(GraphicsContext,Point,int,int,MovingObject):void
-directionState
0..1
<<Java Class>>
Flash
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Flash()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
Brave
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Brave()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
NormalHero
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
NormalHero()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void
<<Java Class>>
Hulk
drawables.characters.heros
HEALTH_POINTS: int
MOTION_DELAY: int
WEAPONS_LIMIT: int
Hulk()
move(Command,Maze):boolean
getHeroStartingHealth():int
addWeapon(Weapon):void

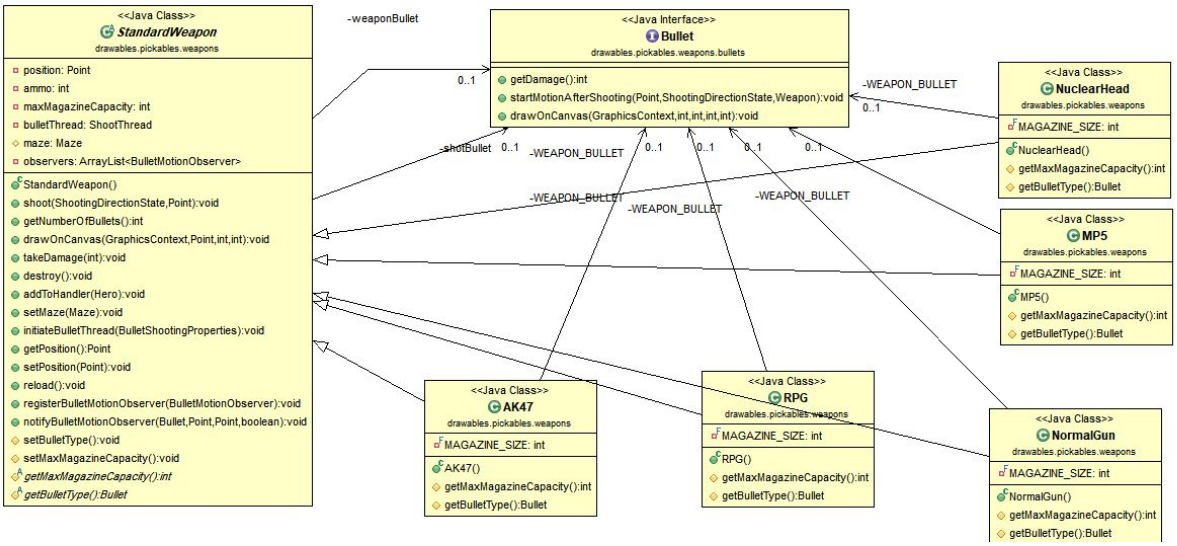

<<Java Class>>
RPGShootingBehavior
drawables.characters.monsters.Behaviors
rifle: Weapon
RPGShootingBehavior()
shoot(Point,DirectionState):void
<<Java Class>>
RifleShootingBehavior
drawables.characters.monsters.Behaviors
rifle: Weapon
RifleShootingBehavior()
shoot(Point,DirectionState):void
<<Java Class>>
NoShootingBehavior
drawables.characters.monsters.Behaviors
NoShootingBehavior()
shoot(Point,DirectionState):void
<<Java Interface>>
ShootingBehavior
drawables.characters.monsters.Behaviors
shoot(Point,DirectionState):void
-shootingBehavior
0..1
<<Java Class>>
StandardMonster
drawables.characters.monsters
damage: int
healthPoints: int
position: Point
pastPosition: Point
maze: Maze
observers: ArrayList<MonsterObserver>
directionState: DirectionState
downSprite: MySprite
upSprite: MySprite
rightSprite: MySprite
leftSprite: MySprite

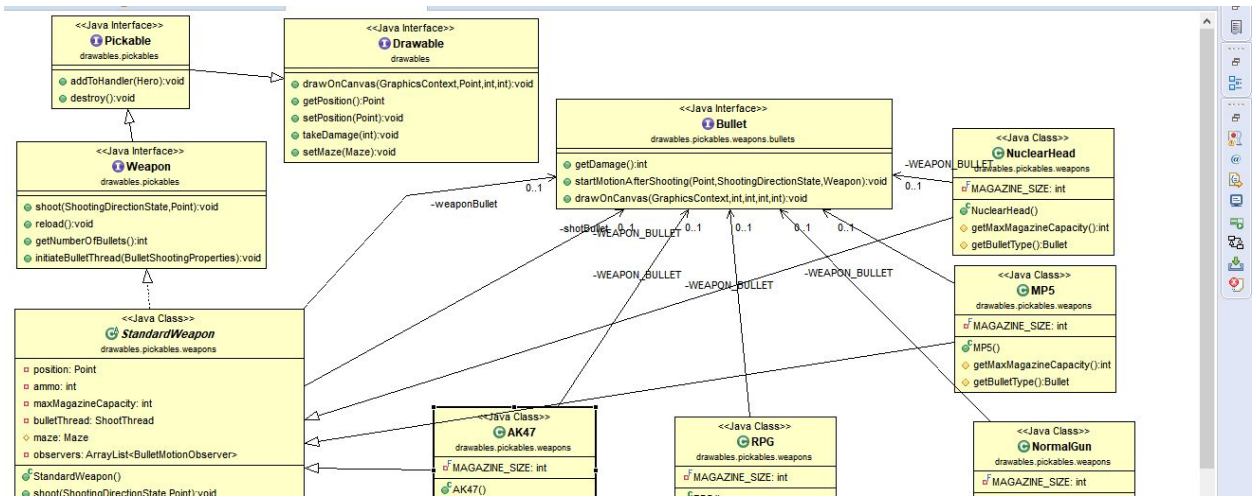

<<Java Class>>
MonstersFactory
drawables.characters.monsters
monsters: ArrayList<Class<? extends Monster>>
MonstersFactory()
getClasses():ArrayList<Class<? extends Monster>>
factoryMethod(String):Monster
loadClasses():boolean
<<Java Class>>
DestroyerMonster
drawables.characters.monsters
HEALTH_POINTS: int
DAMAGE: int
DestroyerMonster()
getMonsterDamage():int
getShootingBehavior():ShootingBehavior
getMonsterHealthPoints():int
<<Java Class>>
NormalMonster
drawables.characters.monsters
HEALTH_POINTS: int
DAMAGE: int
NormalMonster()
getMonsterDamage():int
getShootingBehavior():ShootingBehavior
getMonsterHealthPoints():int
<<Java Class>>
StandardMonster
drawables.characters.monsters
damage: int
healthPoints: int
position: Point
pastPosition: Point
maze: Maze
observers: ArrayList<MonsterObserver>
directionState: DirectionState
downSprite: MySprite
upSprite: MySprite
rightSprite: MySprite
leftSprite: MySprite
StandardMonster()
attack(Hero):void
getDamage():int
setMaze(Maze):void
move(Command,Maze):boolean
getHealthPoints():int
setDirectionState(DirectionState):void
getDirectionState():DirectionState
takeBullet(Bullet):void
receiveExplosionDamage(int):void
takeDamage(int):void
shoot():void
getPosition():Point
getOldPosition():Point
setPosition(Point):void
drawOnCanvas(GraphicsContext,Point,int,int):void
getDownSprite():MySprite
-shootingBehavior
0..1
<<Java Interface>>
ShootingBehavior
drawables.characters.monsters.Behaviors
shoot(Point,DirectionState):void
<<Java Class>>
GiantMonster
drawables.characters.monsters
HEALTH_POINTS: int
DAMAGE: int
GiantMonster()
getMonsterDamage():int
getShootingBehavior():ShootingBehavior
getMonsterHealthPoints():int
<<Java Class>>
ShootingMonster
drawables.characters.monsters
HEALTH_POINTS: int
DAMAGE: int
ShootingMonster()
getMonsterDamage():int
getShootingBehavior():ShootingBehavior
getMonsterHealthPoints():int

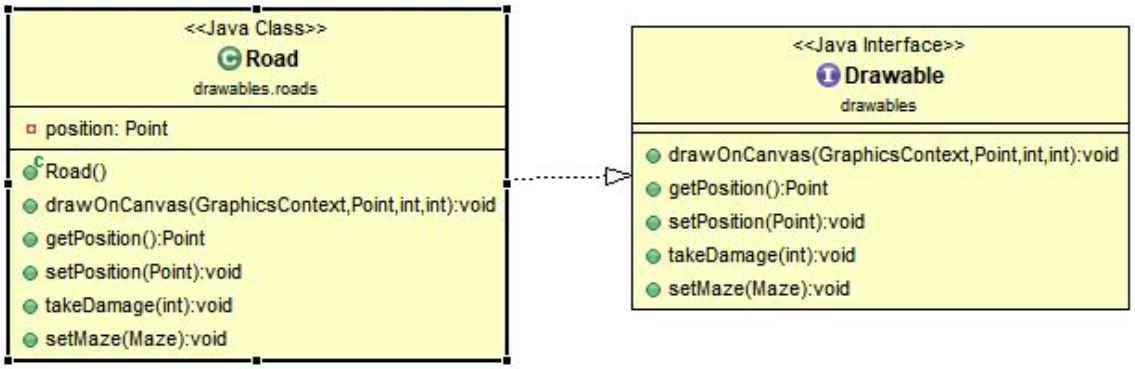

<<Java Class>>
Fire
drawables.obstacles.traps
HEALTH_POINTS: int
DAMAGE: int
Fire()
getHealthPoints():int
getOriginalDamage():int
<<Java Class>>
StandardTrap
drawables.obstacles.traps
position: Point
damage: int
maze: Maze
healthPoints: int
StandardTrap()
getDamage():int
getPosition():Point
setPosition(Point):void
setMaze(Maze):void
destroy():void
trap(Hero):void
takeDamage(int):void
drawOnCanvas(GraphicsContext,Point,int,int):void
setDamage():void
setHealthPoints():void
getOriginalDamage():int
getHealthPoints():int
<<Java Class>>
Hole
drawables.obstacles.traps
HEALTH_POINTS: int
DAMAGE: int
Hole()
getHealthPoints():int
getOriginalDamage():int
<<Java Class>>
RandomTeleport
drawables.obstacles.traps
HEALTH: int
DAMAGE: int
RandomTeleport()
getOriginalDamage():int
getHealthPoints():int
trap(Hero):void
destroy():void
<<Java Class>>
Cage
drawables.obstacles.traps
HEALTH_POINTS: int
DAMAGE: int
Cage()
getHealthPoints():int
trap(Hero):void
getOriginalDamage():int
<<Java Class>>
TrapsFactory
drawables.obstacles.traps
traps: ArrayList<Class<? extends Trap>>
TrapsFactory()
getClasses():ArrayList<Class<? extends Trap>>
factoryMethod(String):Trap
loadClasses():boolean


<<Java Class>>
StandardWall
drawables.obstacles.walls
position: Point
maze: Maze
healthPoints: int
StandardWall()
getPosition():Point
setPosition(Point):void
setMaze(Maze):void
takeDamage(int):void
drawOnCanvas(GraphicsContext,Point,int,int):void
setHealthPoints():void
getHealthPoints():int
<<Java Class>>
Bricks
drawables.obstacles.walls
HEALTH_POINTS: int
Bricks()
getHealthPoints():int
destroy():void
<<Java Interface>>
Wall
drawables.obstacles
<<Java Interface>>
Obstacle
drawables.obstacles
destroy():void
<<Java Interface>>
Drawable
drawables
drawOnCanvas(GraphicsContext,Point,int,int):void
getPosition():Point
setPosition(Point):void
takeDamage(int):void
setMaze(Maze):void
<<Java Class>>
Steel
drawables.obstacles.walls
HEALTH_POINTS: int
Steel()
takeDamage(int):void
getHealthPoints():int
destroy():void
<<Java Class>>
Tree
drawables.obstacles.walls
HEALTH_POINTS: int
Tree()
getHealthPoints():int
destroy():void
<<Java Class>>
Wood
drawables.obstacles.walls
HEALTH_POINTS: int
Wood()
getHealthPoints():int
destroy():void

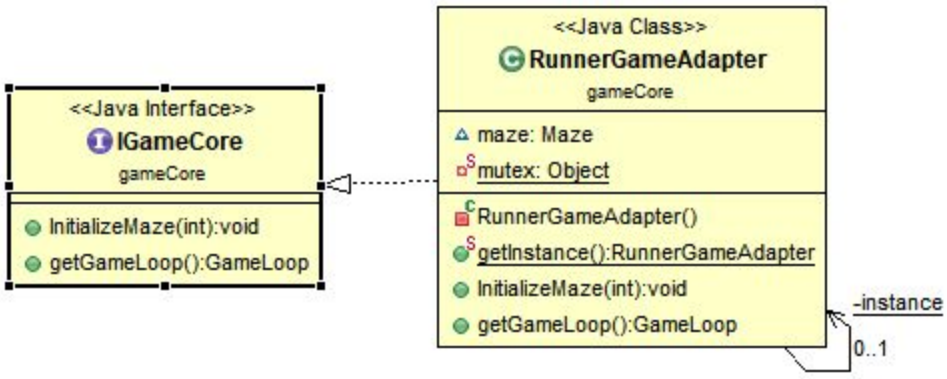

<<Java Class>>
NormalShield
drawables.pickables.shields
SHIELD_POINTS: int
NormalShield()
getShieldPoints():int
<<Java Class>>
PlatinumShield
drawables.pickables.shields
SHIELD_POINTS: int
PlatinumShield()
getShieldPoints():int
<<Java Class>>
SteelShield
drawables.pickables.shields
SHIELD_POINTS: int
SteelShield()
getShieldPoints():int
<<Java Class>>
WoodenShield
drawables.pickables.shields
SHIELD_POINTS: int
WoodenShield()
getShieldPoints():int
<<Java Class>>
StandardShield
drawables.pickables.shields
position: Point
maze: Maze
shieldPoints: int
StandardShield()
addToHandler(Hero):void
setMaze(Maze):void
drawOnCanvas(GraphicsContext,Point,int,int):void
takeDamage(int):void
destroy():void
getPosition():Point
setPosition(Point):void
setShieldPoints():void
addArmorPoints(Hero):void
getShieldPoints():int
<<Java Interface>>
Shield
drawables.pickables
addArmorPoints(Hero):void
<<Java Interface>>
Pickable
drawables.pickables
addToHandler(Hero):void
destroy():void
<<Java Interface>>
Drawable
drawables
drawOnCanvas(GraphicsContext,Point,
getPosition():Point
setPosition(Point):void
takeDamage(int):void
setMaze(Maze):void


<<Java Class>>
StandardWeapon
drawables.pickables.weapons
position: Point
ammo: int
maxMagazineCapacity: int
bulletThread: ShootThread
maze: Maze
observers: ArrayList<BulletMotionObserver>
StandardWeapon()
shoot(ShootingDirectionState,Point):void
getNumberOfBullets():int
drawOnCanvas(GraphicsContext,Point,int,int):void
takeDamage(int):void
destroy():void
addToHandler(Hero):void
setMaze(Maze):void
initiateBulletThread(BulletShootingProperties):void
getPosition():Point
setPosition(Point):void
reload():void
registerBulletMotionObserver(BulletMotionObserver):void
notifyBulletMotionObserver(Bullet,Point,Point,boolean):void
setBulletType():void
setMaxMagazineCapacity():void
getMaxMagazineCapacity():int
getBulletType():Bullet
-weaponBullet
0..1
<<Java Interface>>
Bullet
drawables.pickables.weapons.bullets
getDamage():int
startMotionAfterShooting(Point,ShootingDirectionState,Weapon):void
drawOnCanvas(GraphicsContext,int,int,int,int):void
-WEAPON_BULLET
-shotBullet
<<Java Class>>
NuclearHead
drawables.pickables.weapons
MAGAZINE_SIZE: int
NuclearHead()
getMaxMagazineCapacity():int
getBulletType():Bullet
<<Java Class>>
MP5
drawables.pickables.weapons
MAGAZINE_SIZE: int
MP5()
getMaxMagazineCapacity():int
getBulletType():Bullet
<<Java Class>>
AK47
drawables.pickables.weapons
MAGAZINE_SIZE: int
AK47()
getMaxMagazineCapacity():int
getBulletType():Bullet
<<Java Class>>
RPG
drawables.pickables.weapons
MAGAZINE_SIZE: int
RPG()
getMaxMagazineCapacity():int
getBulletType():Bullet
<<Java Class>>
NormalGun
drawables.pickables.weapons
MAGAZINE_SIZE: int
NormalGun()
getMaxMagazineCapacity():int
getBulletType():Bullet

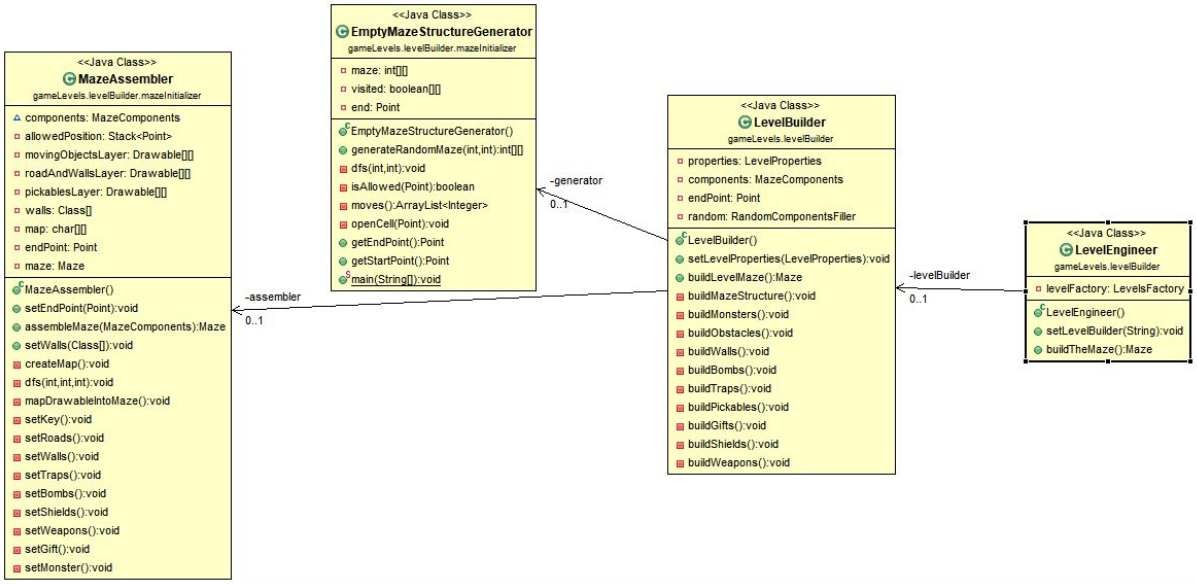

<<Java Interface>>
Pickable
drawables.pickables
addToHandler(Hero):void
destroy():void
<<Java Interface>>
Drawable
drawables
drawOnCanvas(GraphicsContext,Point,int,int):void
getPosition():Point
setPosition(Point):void
takeDamage(int):void
setMaze(Maze):void
<<Java Interface>>
Weapon
drawables.pickables
shoot(ShootingDirectionState,Point):void
reload():void
getNumberOfBullets():int
initiateBulletThread(BulletShootingProperties):void
<<Java Class>>
StandardWeapon
drawables.pickables.weapons
position: Point
ammo: int
maxMagazineCapacity: int
bulletThread: ShootThread
maze: Maze
observers: ArrayList<BulletMotionObserver>
StandardWeapon()
-weaponBullet
0..1
<<Java Interface>>
Bullet
drawables.pickables.weapons.bullets
getDamage():int
startMotionAfterShooting(Point,ShootingDirectionState,Weapon):void
drawOnCanvas(GraphicsContext,int,int,int,int):void
-WEAPON_BULLET
-shotBullet
<<Java Class>>
NuclearHead
drawables.pickables.weapons
MAGAZINE_SIZE: int
NuclearHead()
getMaxMagazineCapacity():int
getBulletType():Bullet
<<Java Class>>
MP5
drawables.pickables.weapons
MAGAZINE_SIZE: int
MP5()
getMaxMagazineCapacity():int
getBulletType():Bullet
<<Java Class>>
AK47
drawables.pickables.weapons
MAGAZINE_SIZE: int
<<Java Class>>
RPG
drawables.pickables.weapons
MAGAZINE_SIZE: int
<<Java Class>>
NormalGun
drawables.pickables.weapons
MAGAZINE_SIZE: int

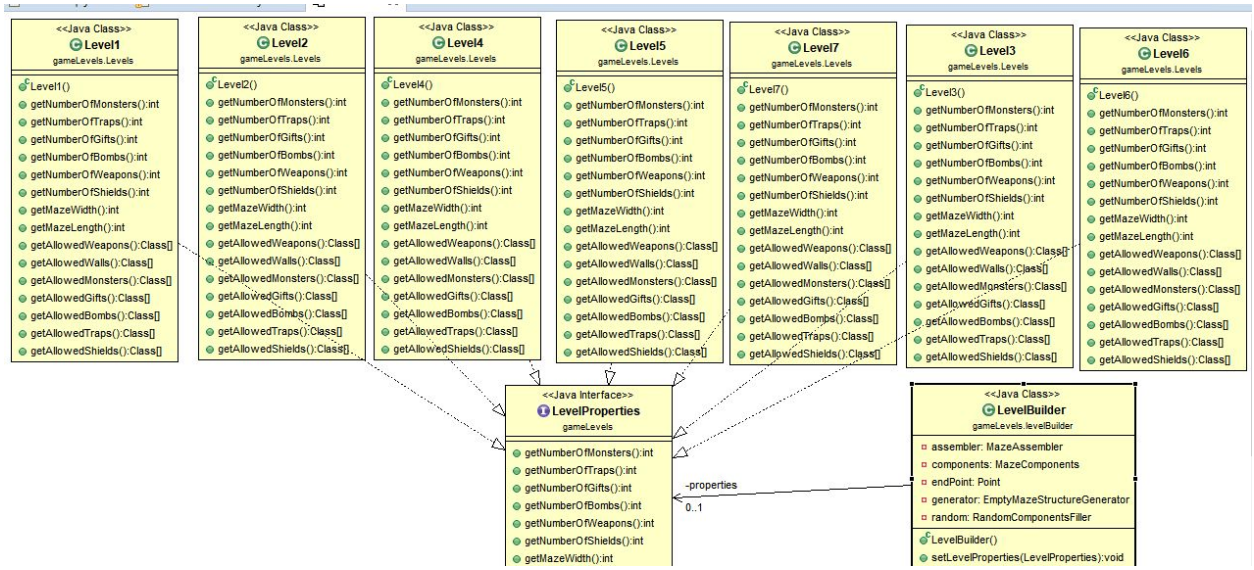


b. Game Core:

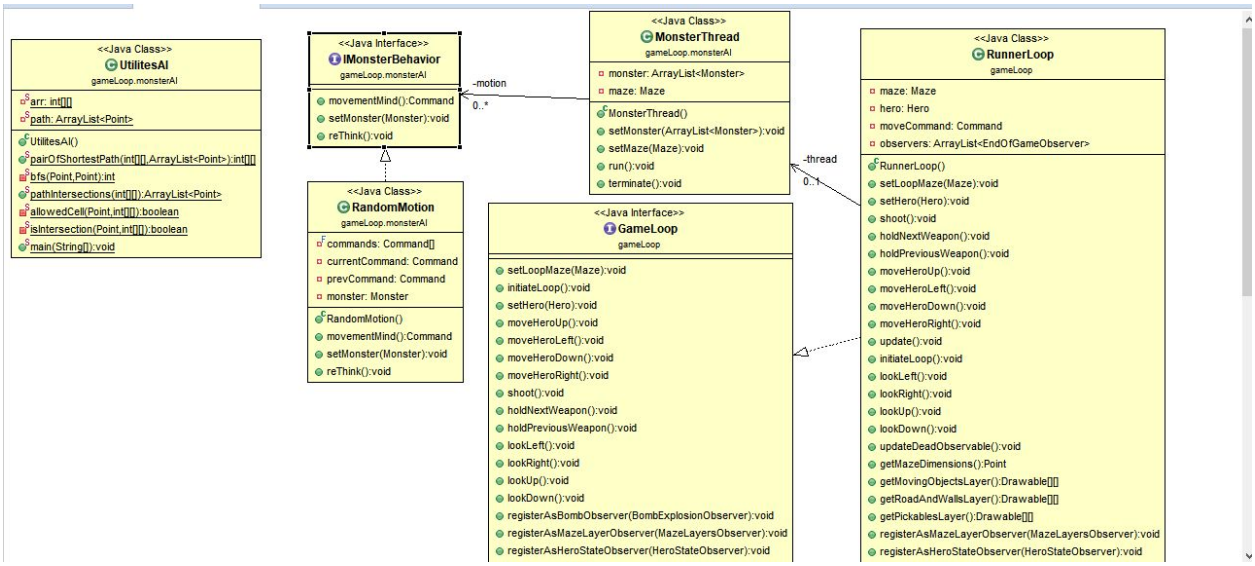


c. Game Level:


<<Java Class>>
MazeAssembler
gameLevels.levelBuilder.mazeInitializer
components: MazeComponents
allowedPosition: Stack<Point>
movingObjectsLayer: Drawable[][]
roadAndWallsLayer: Drawable[][]
pickablesLayer: Drawable[][]
walls: Class[]
map: char[][]
endPoint: Point
maze: Maze
MazeAssembler()
setEndPoint(Point):void
assembleMaze(MazeComponents):Maze
setWalls(Class[]):void
createMap():void
dfs(int,int,int):void
mapDrawableIntoMaze():void
setKey():void
setRoads():void
setWalls():void
setTraps():void
setBombs():void
setShields():void
setWeapons():void
setGift():void
setMonster():void
<<Java Class>>
EmptyMazeStructureGenerator
gameLevels.levelBuilder.mazeInitializer
maze: int[][]
visited: boolean[][]
end: Point
EmptyMazeStructureGenerator()
generateRandomMaze(int,int):int[][]
dfs(int,int):void
isAllowed(Point):boolean
moves():ArrayList<Integer>
openCell(Point):void
getEndPoint():Point
getStartPoint():Point
main(String[]):void
-generator
0..1
-assembler
0..1
<<Java Class>>
LevelBuilder
gameLevels.levelBuilder
properties: LevelProperties
components: MazeComponents
endPoint: Point
random: RandomComponentsFiller
LevelBuilder()
setLevelProperties(LevelProperties):void
buildLevelMaze():Maze
buildMazeStructure():void
buildMonsters():void
buildObstacles():void
buildWalls():void
buildBombs():void
buildTraps():void
buildPickables():void
buildGifts():void
buildShields():void
buildWeapons():void
-levelBuilder
0..1
<<Java Class>>
LevelEngineer
gameLevels.levelBuilder
levelFactory: LevelsFactory
LevelEngineer()
setLevelBuilder(String):void
buildTheMaze():Maze

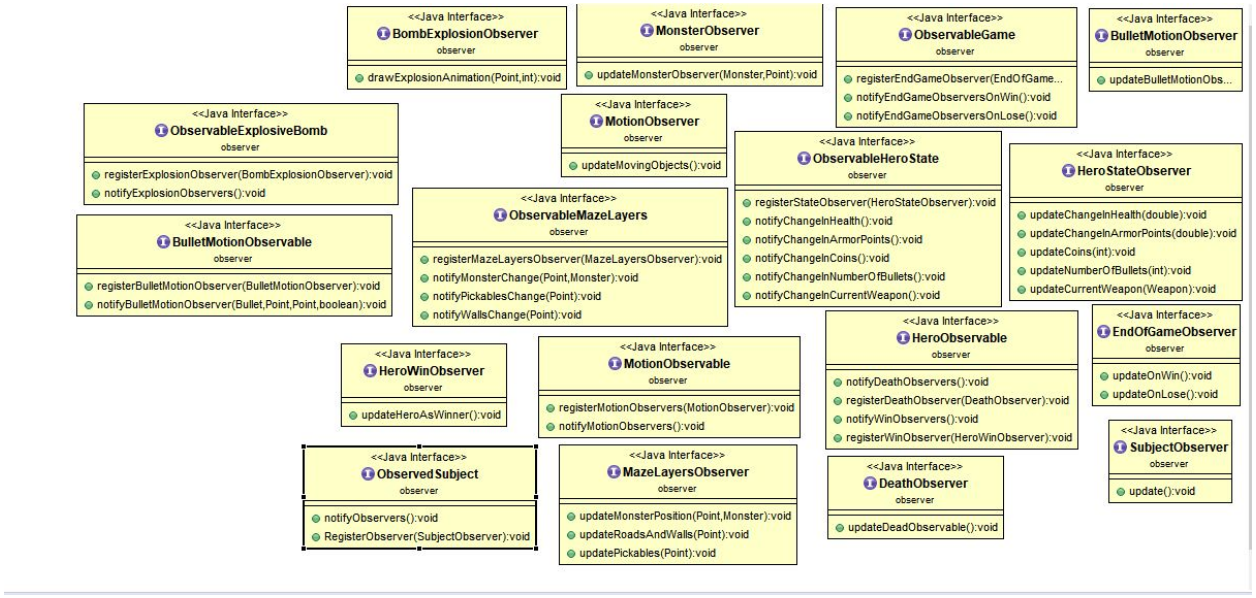

<<Java Class>>
Level1
gameLevels.Levels
Level1()
getNumberOfMonsters():int
getNumberOfTraps():int
getNumberOfGifts():int
getNumberOfBombs():int
getNumberOfWeapons():int
getNumberOfShields():int
getMazeWidth():int
getMazeLength():int
getAllowedWeapons():Class[]
getAllowedWalls():Class[]
getAllowedMonsters():Class[]
getAllowedGifts():Class[]
getAllowedBombs():Class[]
getAllowedTraps():Class[]
getAllowedShields():Class[]
<<Java Class>>
Level2
gameLevels.Levels
Level2()
<<Java Class>>
Level4
gameLevels.Levels
Level4()
<<Java Class>>
Level5
gameLevels.Levels
Level5()
<<Java Class>>
Level7
gameLevels.Levels
Level7()
<<Java Class>>
Level3
gameLevels.Levels
Level3()
<<Java Class>>
Level6
gameLevels.Levels
Level6()
<<Java Interface>>
LevelProperties
gameLevels
<<Java Class>>
LevelBuilder
gameLevels.levelBuilder
assembler: MazeAssembler
components: MazeComponents
endPoint: Point
generator: EmptyMazeStructureGenerator
random: RandomComponentsFiller
LevelBuilder()
setLevelProperties(LevelProperties):void
-properties
0..1

d. Game Loop:

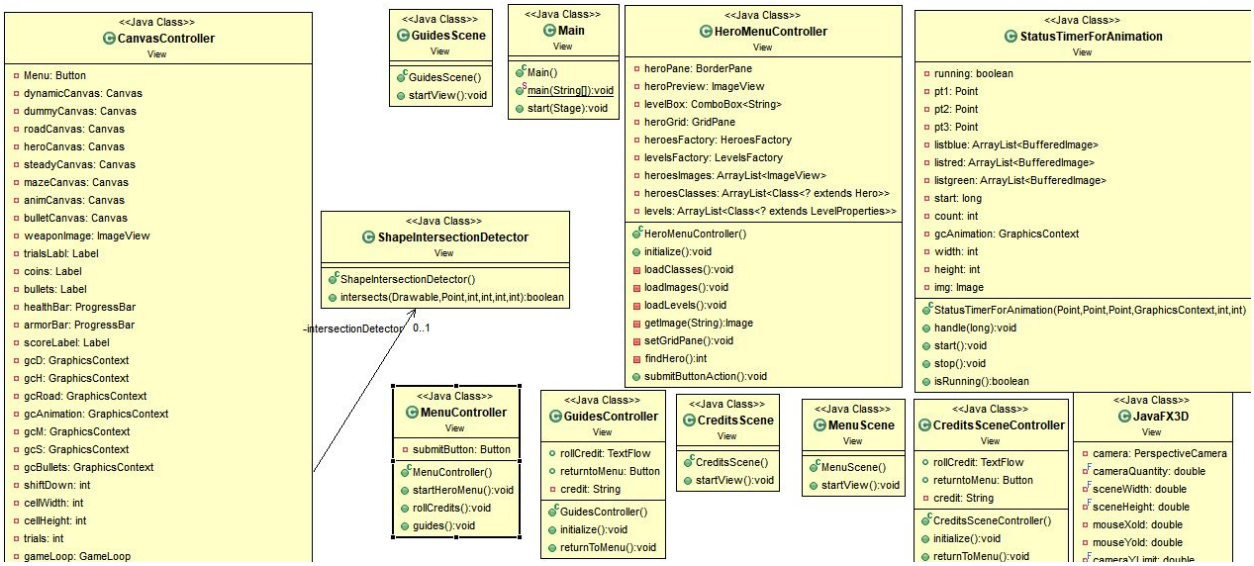


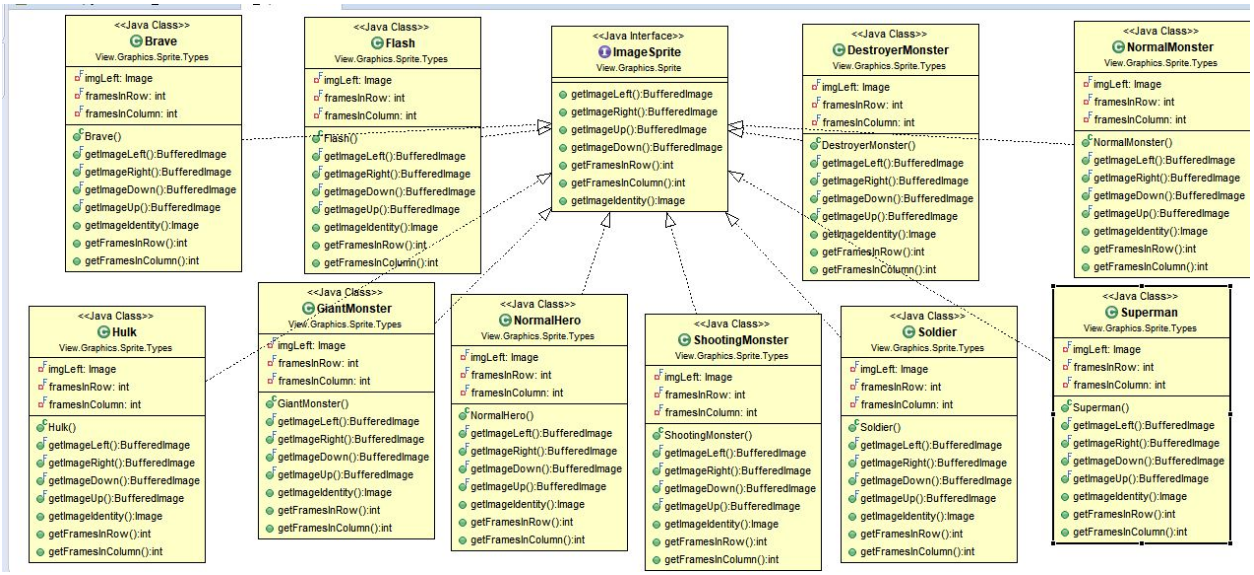
e. Maze :

f. Observer:

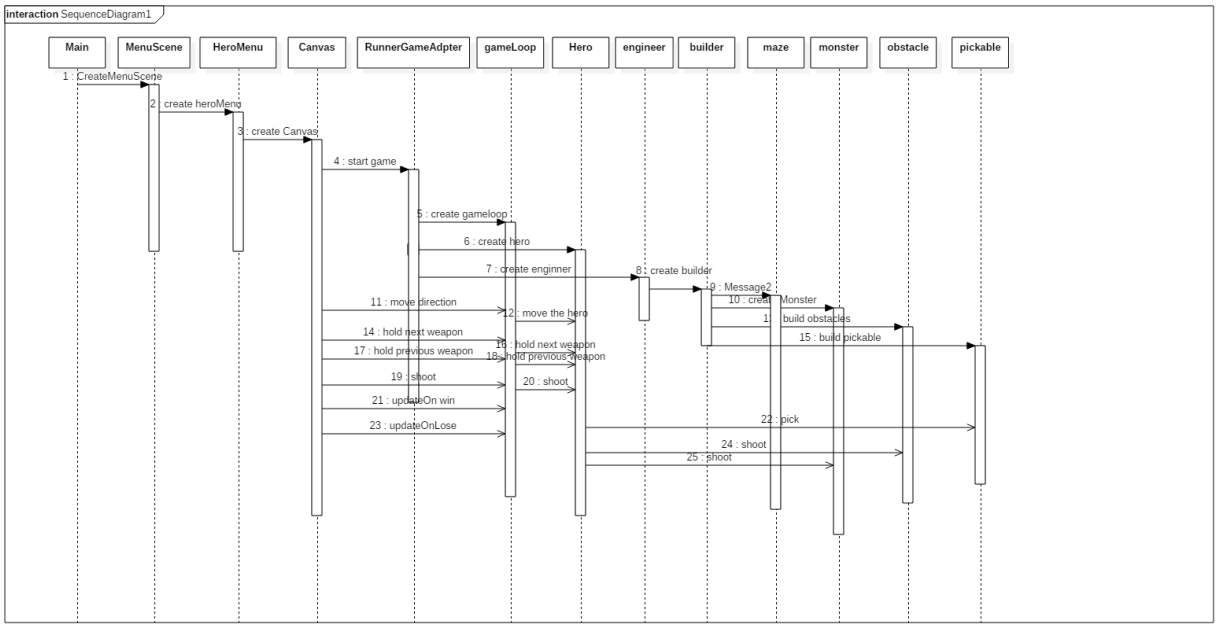

<<Java Interface>>
BombExplosionObserver
observer
drawExplosionAnimation(Point,int):void
<<Java Interface>>
MonsterObserver
observer
updateMonsterObserver(Monster,Point):void
<<Java Interface>>
ObservableGame
observer
registerEndGameObserver(EndOfGame...
notifyEndGameObserversOnWin():void
notifyEndGameObserversOnLose():void
<<Java Interface>>
BulletMotionObserver
observer
updateBulletMotionObs...
<<Java Interface>>
ObservableExplosiveBomb
observer
registerExplosionObserver(BombExplosionObserver):void
notifyExplosionObservers():void
<<Java Interface>>
MotionObserver
observer
updateMovingObjects():void
<<Java Interface>>
ObservableHeroState
observer
registerStateObserver(HeroStateObserver):void
notifyChangeInHealth():void
notifyChangeInArmorPoints():void
notifyChangeInCoins():void
notifyChangeInNumberOfBullets():void
notifyChangeInCurrentWeapon():void
<<Java Interface>>
HeroStateObserver
observer
updateChangeInHealth(double):void
updateChangeInArmorPoints(double):void
updateCoins(int):void
updateNumberOfBullets(int):void
updateCurrentWeapon(Weapon):void
<<Java Interface>>
BulletMotionObservable
observer
registerBulletMotionObserver(BulletMotionObserver):void
notifyBulletMotionObserver(Bullet,Point,Point,boolean):void
<<Java Interface>>
ObservableMazeLayers
observer
registerMazeLayersObserver(MazeLayersObserver):void
notifyMonsterChange(Point,Monster):void
notifyPickablesChange(Point):void
notifyWallsChange(Point):void
<<Java Interface>>
HeroObservable
observer
notifyDeathObservers():void
registerDeathObserver(DeathObserver):void
notifyWinObservers():void
registerWinObserver(HeroWinObserver):void
<<Java Interface>>
EndOfGameObserver
observer
updateOnWin():void
updateOnLose():void
<<Java Interface>>
HeroWinObserver
observer
updateHeroAsWinner():void
<<Java Interface>>
MotionObservable
observer
registerMotionObservers(MotionObserver):void
notifyMotionObservers():void
<<Java Interface>>
SubjectObserver
observer
update():void
<<Java Interface>>
ObservedSubject
observer
notifyObservers():void
RegisterObserver(SubjectObserver):void
<<Java Interface>>
MazeLayersObserver
observer
updateMonsterPosition(Point,Monster):void
updateRoadsAndWalls(Point):void
updatePickables(Point):void
<<Java Interface>>
DeathObserver
observer
updateDeadObservable():void

g. View:

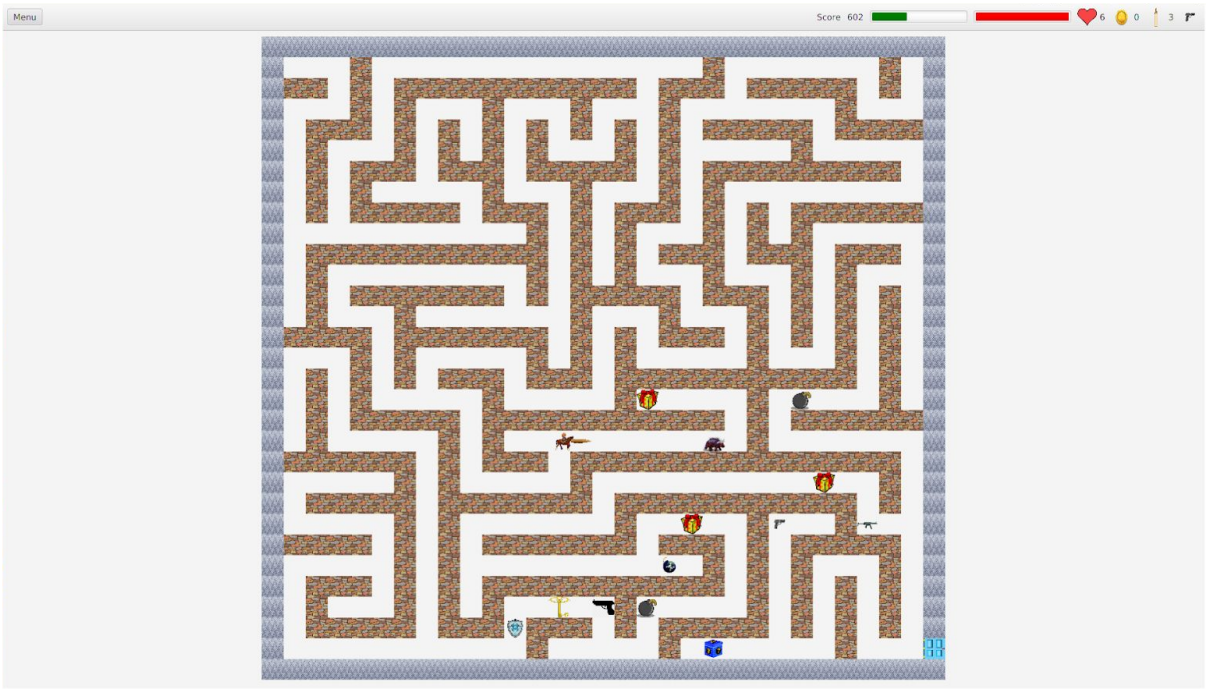

<<Java Class>>
CanvasController
View
Menu: Button
dynamicCanvas: Canvas
dummyCanvas: Canvas
roadCanvas: Canvas
heroCanvas: Canvas
steadyCanvas: Canvas
mazeCanvas: Canvas
animCanvas: Canvas
bulletCanvas: Canvas
weaponImage: ImageView
trialsLabl: Label
coins: Label
bullets: Label
healthBar: ProgressBar
armorBar: ProgressBar
scoreLabel: Label
gcD: GraphicsContext
gcH: GraphicsContext
gcRoad: GraphicsContext
gcAnimation: GraphicsContext
gcM: GraphicsContext
gcS: GraphicsContext
gcBullets: GraphicsContext
shiftDown: int
cellWidth: int
cellHeight: int
trials: int
gameLoop: GameLoop
<<Java Class>>
GuidesScene
View
GuidesScene()
startView():void
<<Java Class>>
Main
View
Main()
main(String[]):void
start(Stage):void
<<Java Class>>
HeroMenuController
View
heroPane: BorderPane
heroPreview: ImageView
levelBox: ComboBox<String>
heroGrid: GridPane
heroesFactory: HeroesFactory
levelsFactory: LevelsFactory
heroesImages: ArrayList<ImageView>
heroesClasses: ArrayList<Class<? extends Hero>>
levels: ArrayList<Class<? extends LevelProperties>>
HeroMenuController()
initialize():void
loadClasses():void
loadImages():void
loadLevels():void
getImage(String):Image
setGridPane():void
findHero():int
submitButtonAction():void
<<Java Class>>
StatusTimerForAnimation
View
running: boolean
pt1: Point
pt2: Point
pt3: Point
listblue: ArrayList<BufferedImage>
listred: ArrayList<BufferedImage>
listgreen: ArrayList<BufferedImage>
start: long
count: int
gcAnimation: GraphicsContext
width: int
height: int
img: Image
StatusTimerForAnimation(Point,Point,Point,GraphicsContext,int,int)
handle(long):void
start():void
stop():void
isRunning():boolean
<<Java Class>>
ShapeIntersectionDetector
View
ShapeIntersectionDetector()
intersects(Drawable,Point,int,int,int,int):boolean
-intersectionDetector
0..1
<<Java Class>>
MenuController
View
submitButton: Button
MenuController()
startHeroMenu():void
rollCredits():void
guides():void
<<Java Class>>
GuidesController
View
rollCredit: TextFlow
returntoMenu: Button
credit: String
GuidesController()
initialize():void
returnToMenu():void
<<Java Class>>
CreditsScene
View
CreditsScene()
startView():void
<<Java Class>>
MenuScene
View
MenuScene()
startView():void
<<Java Class>>
CreditsSceneController
View
rollCredit: TextFlow
returntoMenu: Button
credit: String
CreditsSceneController()
initialize():void
returnToMenu():void
<<Java Class>>
JavaFX3D
View
camera: PerspectiveCamera
cameraQuantity: double
sceneWidth: double
sceneHeight: double
mouseXold: double
mouseYold: double

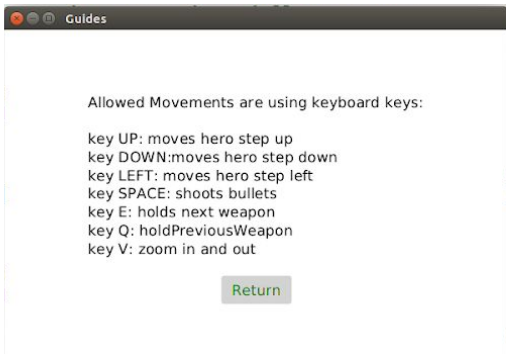


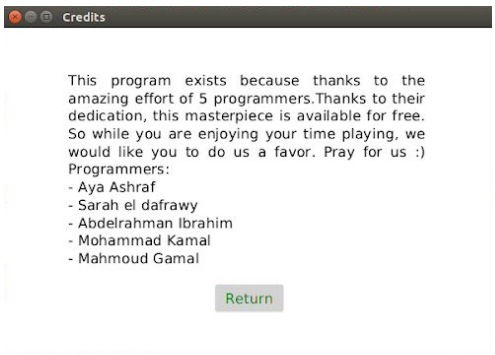
2. Sequence Diagram :



GUI Snapshots:


Menu
Score 602


Guides
Allowed Movements are using keyboard keys:
key UP: moves hero step up
key DOWN:moves hero step down
key LEFT: moves hero step left
key SPACE: shoots bullets
key E: holds next weapon
key Q: holdPreviousWeapon
key V: zoom in and out
Return


Credits
This program exists because thanks to the amazing effort of 5 programmers.Thanks to their dedication, this masterpiece is available for free. So while you are enjoying your time playing, we would like you to do us a favor. Pray for us :)
Programmers:
- Aya Ashraf
- Sarah el dafrawy
- Abdelrahman Ibrahim
- Mohammad Kamal
- Mahmoud Gamal
Return