COMP9003
SID: 540263642

The game was built using a combination of classes, abstract class, and interface.

App class is used as the main class and is used to mainly setup all the resources required for the game, draws the level continuously, updates the UI, checking player and game state, and processing input from players.

Player class is used to track the players score, powerups, and colours. It is created as a separate class as the player information persists across each level depending on the tank they are assigned to. The player information is very easy to reset when the game ends as it is disjoint from the levels.

Level class is used to create the playing level from the level text file provided, rendering the terrain and every **LevelObject** a level contains (tree, tank, projectile). The level class does so by keeping track of every LevelObject through a simple array (which gets continuously updated) and each level has their own LevelObjects. It is also used to set the wind at the start of every level.

LevelObject abstract class is used to help managing every object in a level. Since every object in a level has an (x,y) coordinate, needs to be drawn, can be active or inactive, and/or can be belong to a player or not, we create an abstract class for every level objects to be inherited from to reduce the number common parameters and methods. Once a LevelObject is inactive, we stop drawing it.

Tree class (extends LevelObject class) is used to draw the tree at a certain position (x,y) and falls if the terrain level is different from the (y) value of the tree. It is initialized using its (x,y) position and the level it belongs to.

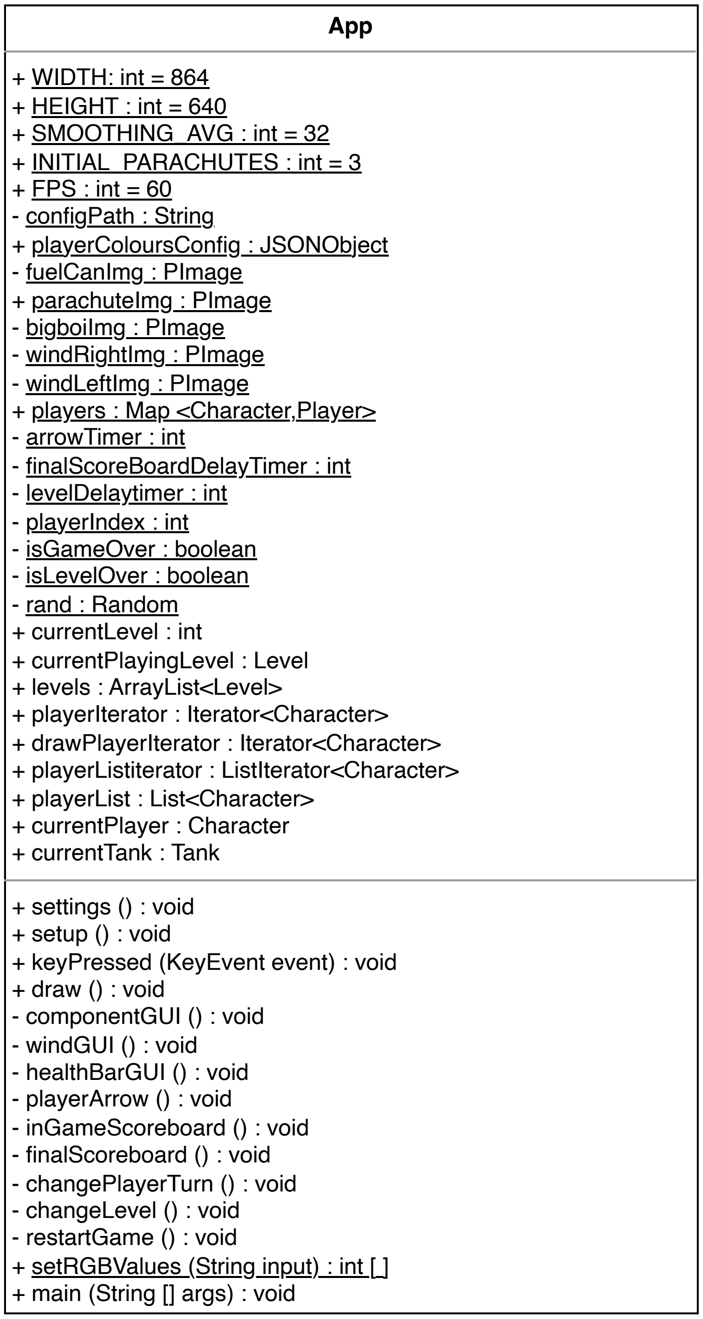
Tank class (extends LevelObject class and implements **Explosion**) is used to track the movement of the tank (user input, fuel consumption, fall and fall damage). We separate their movement from projectiles as they move differently. It is initialized by using the level it belongs to, a starting position and a Player object which the tank belongs to. It also implements the Explosion interface discussed below.

Projectile class (extends LevelObject class and implements **Explosion**) is used to track the trajectory of a projectile from a Tank. Once fired, the move method is repeatedly drawn until a collision happens. The projectile is initialized using the Level it belongs to, the Tank that fired it, and the wind at that moment. Thus, at collision, the projectile explosion is used to level the terrain. It also implements the Explosion interface discussed below.

Explosion interface is used to define the explosion animation which occurs in both Tank and Projectile. We use the interface here to reduce the amount of repetitive code to draw the explosion. It is implemented by using the gradual increase of radius of the explosion.

<u>Extension Implemented</u>

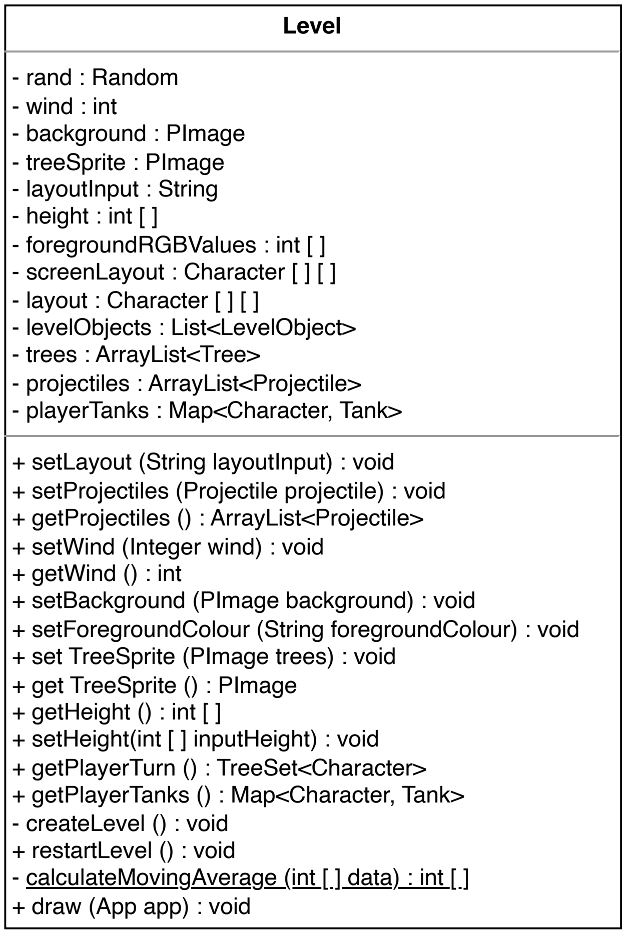
Name	Description	Key
Repair Kit	A player can trade 20 of its point for 20 health points	R
Additional Fuel	A player can trade 10 of its points for 200 fuel points (fuel cap at 250)	F
Additional Parachute	A player can trade 15 of its points for an additional parachute	P
Larger Projectile	A player can trade 20 of its points to shoot a larger projectile (twice the normal size but with same damage)	X



App

```
+ WIDTH: int = 864
+ HEIGHT : int = 640
+ SMOOTHING_AVG : int = 32
+ INITIAL_PARACHUTES : int = 3
+ FPS : int = 60
- configPath : String
+ playerColoursConfig : JSONObject
- fuelCanImg : PImage
+ parachuteImg : PImage
- bigboiImg : PImage
- windRightImg : PImage
- windLeftImg : PImage
+ players : Map <Character,Player>
- arrowTimer : int
- finalScoreBoardDelayTimer : int
- levelDelaytimer : int
- playerIndex : int
- isGameOver : boolean
- isLevelOver : boolean
- rand : Random
+ currentLevel : int
+ currentPlayingLevel : Level
+ levels : ArrayList<Level>
+ playerIterator : Iterator<Character>
+ drawPlayerIterator : Iterator<Character>
+ playerListiterator : ListIterator<Character>
+ playerList : List<Character>
+ currentPlayer : Character
+ currentTank : Tank
```

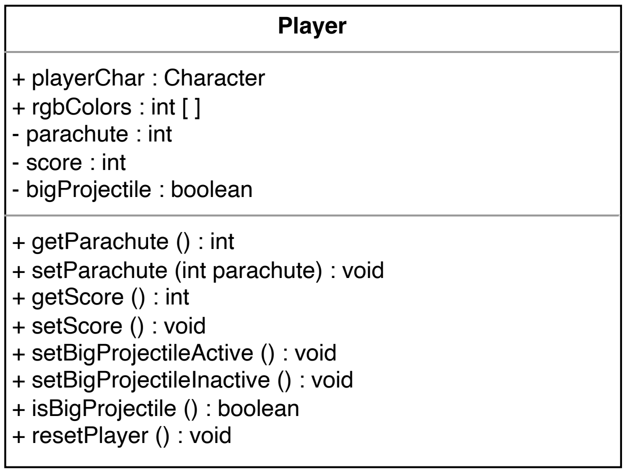
```
+ settings () : void
+ setup () : void
+ keyPressed (KeyEvent event) : void
+ draw () : void
- componentGUI () : void
- windGUI () : void
- healthBarGUI () : void
- playerArrow () : void
- inGameScoreboard () : void
- finalScoreboard () : void
- changePlayerTurn () : void
- changeLevel () : void
- restartGame () : void
+ setRGBValues (String input) : int [ ]
+ main (String [] args) : void
```



Level

```
- rand : Random
- wind : int
- background : PImage
- treeSprite : PImage
- layoutInput : String
- height : int [ ]
- foregroundRGBValues : int [ ]
- screenLayout : Character [ ] [ ]
- layout : Character [ ] [ ]
- levelObjects : List<LevelObject>
- trees : ArrayList<Tree>
- projectiles : ArrayList<Projectile>
- playerTanks : Map<Character, Tank>
```

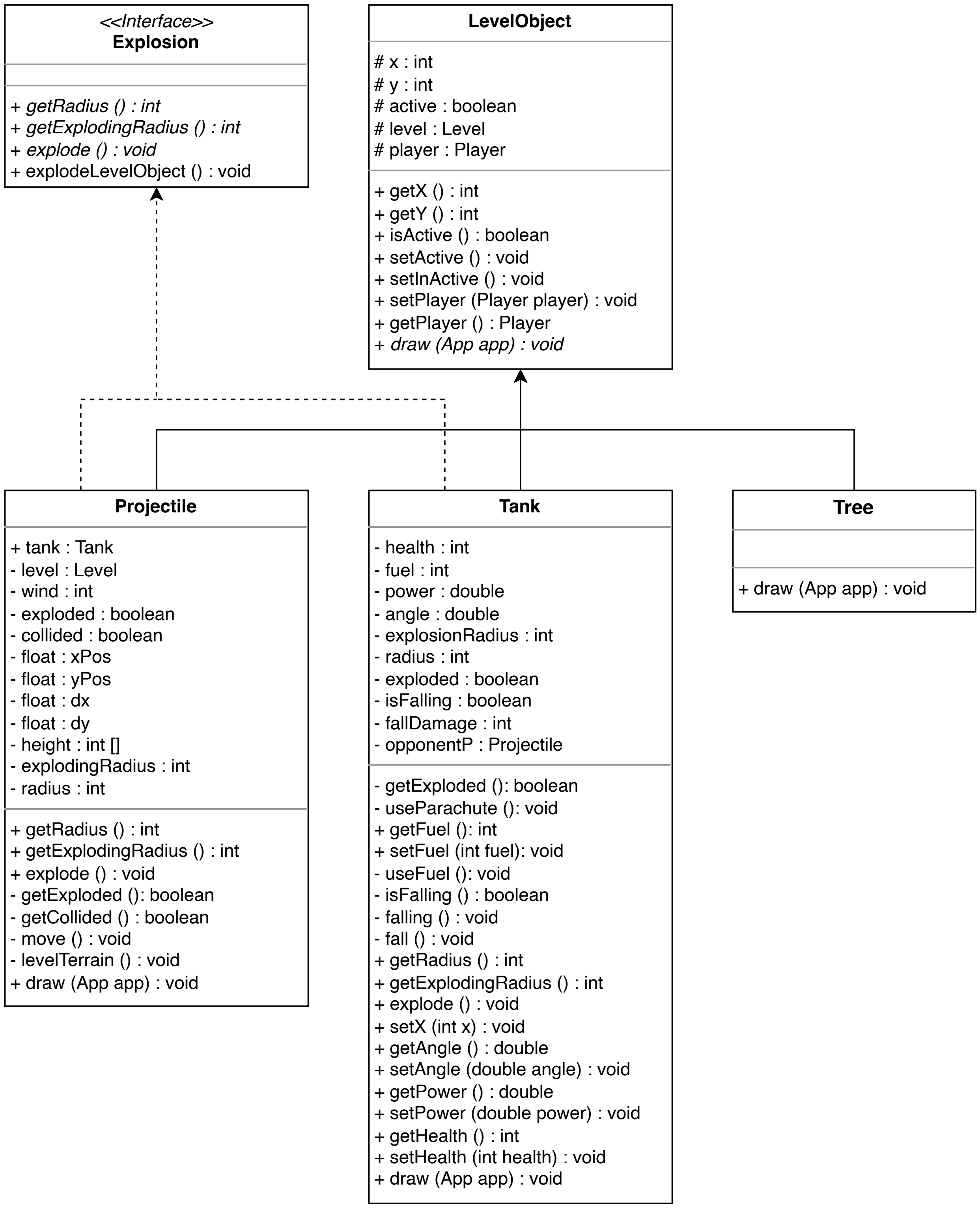
```
+ setLayout (String layoutInput) : void
+ setProjectiles (Projectile projectile) : void
+ getProjectiles () : ArrayList<Projectile>
+ setWind (Integer wind) : void
+ getWind () : int
+ setBackground (PImage background) : void
+ setForegroundColour (String foregroundColour) : void
+ set TreeSprite (PImage trees) : void
+ get TreeSprite () : PImage
+ getHeight () : int [ ]
+ setHeight(int [ ] inputHeight) : void
+ getPlayerTurn () : TreeSet<Character>
+ getPlayerTanks () : Map<Character, Tank>
- createLevel () : void
+ restartLevel () : void
- calculateMovingAverage (int [ ] data) : int [ ]
+ draw (App app) : void
```



Player

```
+ playerChar : Character
+ rgbColors : int [ ]
- parachute : int
- score : int
- bigProjectile : boolean
```

```
+ getParachute () : int
+ setParachute (int parachute) : void
+ getScore () : int
+ setScore () : void
+ setBigProjectileActive () : void
+ setBigProjectileInactive () : void
+ isBigProjectile () : boolean
+ resetPlayer () : void
```



<<Interface>>
Explosion
+ getRadius () : int
+ getExplodingRadius () : int
+ explode () : void
+ explodeLevelObject () : void
LevelObject
# x : int
# y : int
# active : boolean
# level : Level
# player : Player
+ getX () : int
+ getY () : int
+ isActive () : boolean
+ setActive () : void
+ setInActive () : void
+ setPlayer (Player player) : void
+ getPlayer () : Player
+ draw (App app) : void
Projectile
+ tank : Tank
- level : Level
- wind : int
- exploded : boolean
- collided : boolean
- float : xPos
- float : yPos
- float : dx
- float : dy
- height : int []
- explodingRadius : int
- radius : int
+ getRadius () : int
+ getExplodingRadius () : int
+ explode () : void
- getExploded (): boolean
- getCollided () : boolean
- move () : void
- levelTerrain () : void
+ draw (App app) : void
Tank
- health : int
- fuel : int
- power : double
- angle : double
- explosionRadius : int
- radius : int
- exploded : boolean
- isFalling : boolean
- fallDamage : int
- opponentP : Projectile
- getExploded (): boolean
- useParachute (): void
+ getFuel (): int
+ setFuel (int fuel): void
- useFuel (): void
- isFalling () : boolean
- falling () : void
- fall () : void
+ getRadius () : int
+ getExplodingRadius () : int
+ explode () : void
+ setX (int x) : void
+ getAngle () : double
+ setAngle (double angle) : void
+ getPower () : double
+ setPower (double power) : void
+ getHealth () : int
+ setHealth (int health) : void
+ draw (App app) : void
Tree
+ draw (App app) : void