# *CSC 413 Project Documentation*

# *Fall 2018*


***Amir R. Anjomshoaa***

***918710278***

***Section 01 (MW 2:30—4:25PM)***


***https://github.com/csc413-01-summer2019/csc413-tankgame-amiranjom.git***

# Table of Contents

# 1 Introduction

## 1.1 Project Overview

This project came out to be an amazing project in the process. The project main concept is the idea of OOP and abstracting the objects. We would be designing a game throughout this project. The OOP design in this project is the main part of our focus and designing all.

## 1.2 Introduction of the Tank Game

The Game is a basic Tank game. The tank would be respawning in 2 different locations in separate windows. Each player should try to outlast the other player by shooting rockets at each other. Each player starts with the health of 100 and total life of 3. Life and health is available to be picked up throughout the game. The game also involves some unbreakable box to stop the player to go everywhere but there is also breakable box which could give the attacker advantage point.

# 2 Development Environment

In this assignment I used JDK 11 version. And the IDE I used was IntelliJ due to the simple environment it has.

My main special resource would be a series of video I watched on YouTube a tutorial for game development.

https://www.youtube.com/playlist?list=PLWms45O3n--5vDnNd6aiu1CSWX3JlCU1n

# 3 How to Build/Import your Project

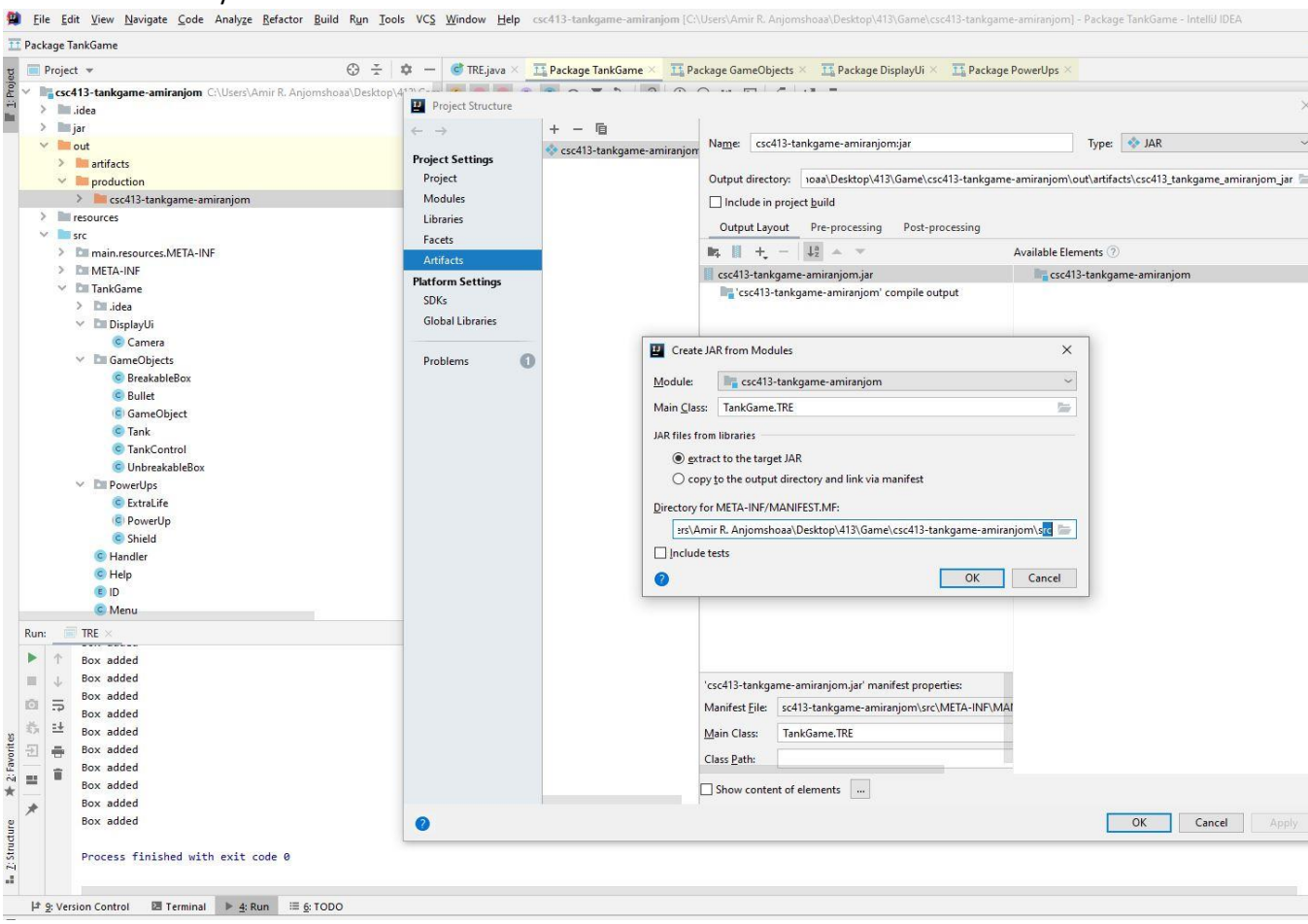
## 3.1 Import project (Intellij)

1) Clone the program from github into the specific folder you want.
2) Open up intellij and click on the import project.
3) Import the project from the csc413-tankgame-amiranjom folder (main root)
4) After importing by building the project you are able to run the game from the TRE main function

## 3.2 .Jar File (Intellij) and Dependency (Resource Folder needs to be added)

1) Go to the project structure tab Ctrl+Alt+Shift+s
2) Page opens up on the left hand side click Artifacts under project setting
3) In the Artifacts page, on the left it should be a plus “+” sign click on that
4) Choose the option JAR -> From Module Dependency
5) A page opens up after, in the section that says Main class put “TankGame.TRE” and for

META-INF directory choose the src folder




6) After that save everything come back to the main page. Click on Build on the top -> Build Artifacts -> rebuild
7) Afterwards inside the output folder there should be artifacts folder and inside that a .JAR file
8) Lastly for making the .JAR work you need to copy my resource folder with the name resources and copy it next to the .Jar file in the same file and this should allow you to run my .JAR file.

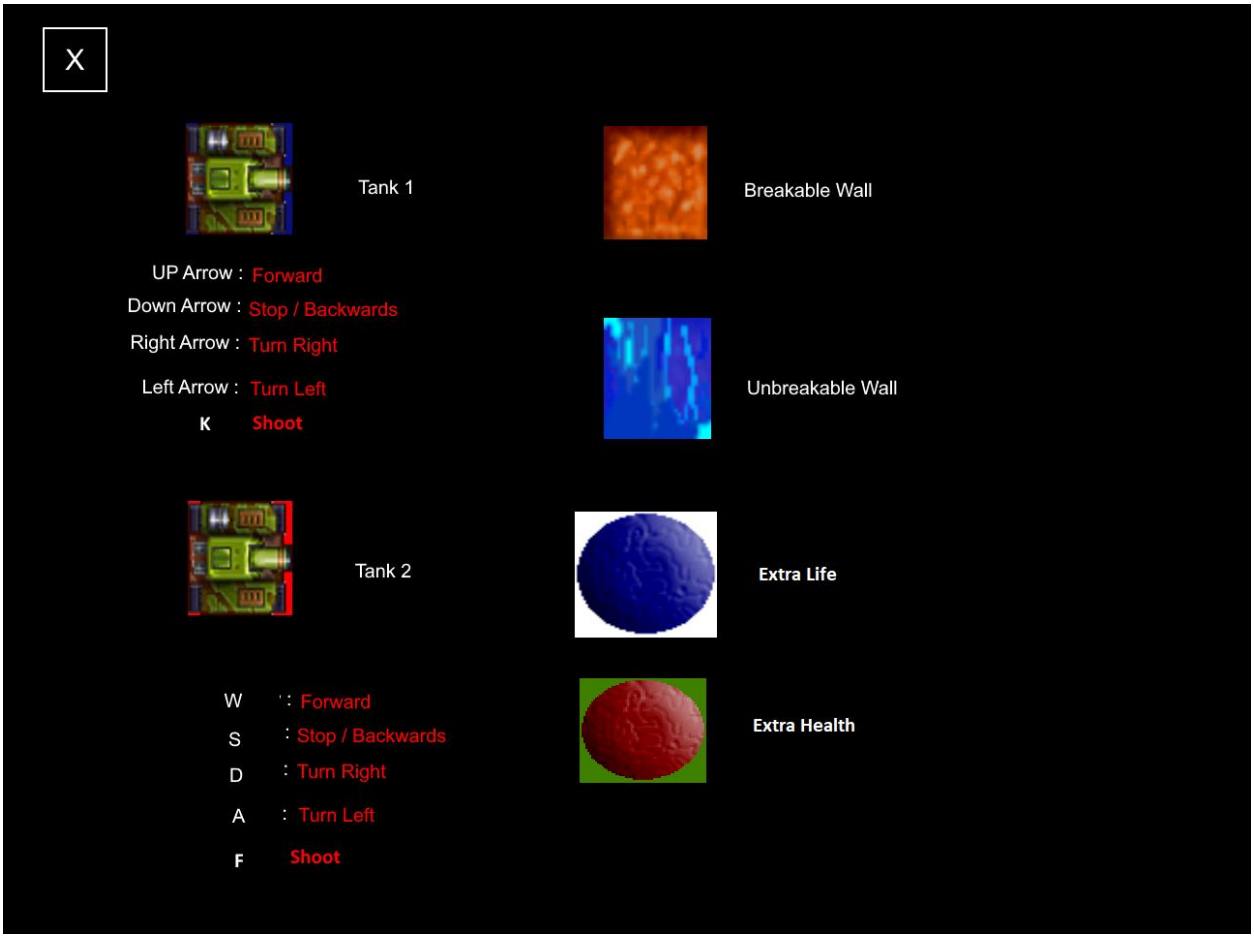
# 4 How to Run your Game

## 4.1 Run the Game

At this time, you should have the game already built inside intellij and you just need to run the main function inside the TRE class, and also you should have the .Jar file ready by having the resource folder next to the .Jar file you should be able to run the game

## 4.2 Controls

I would lay out the controls for the game over here but also inside my game at the beginning there should be a help option that would show the full info for the game and everything related to it.




# 5 Assumption Made

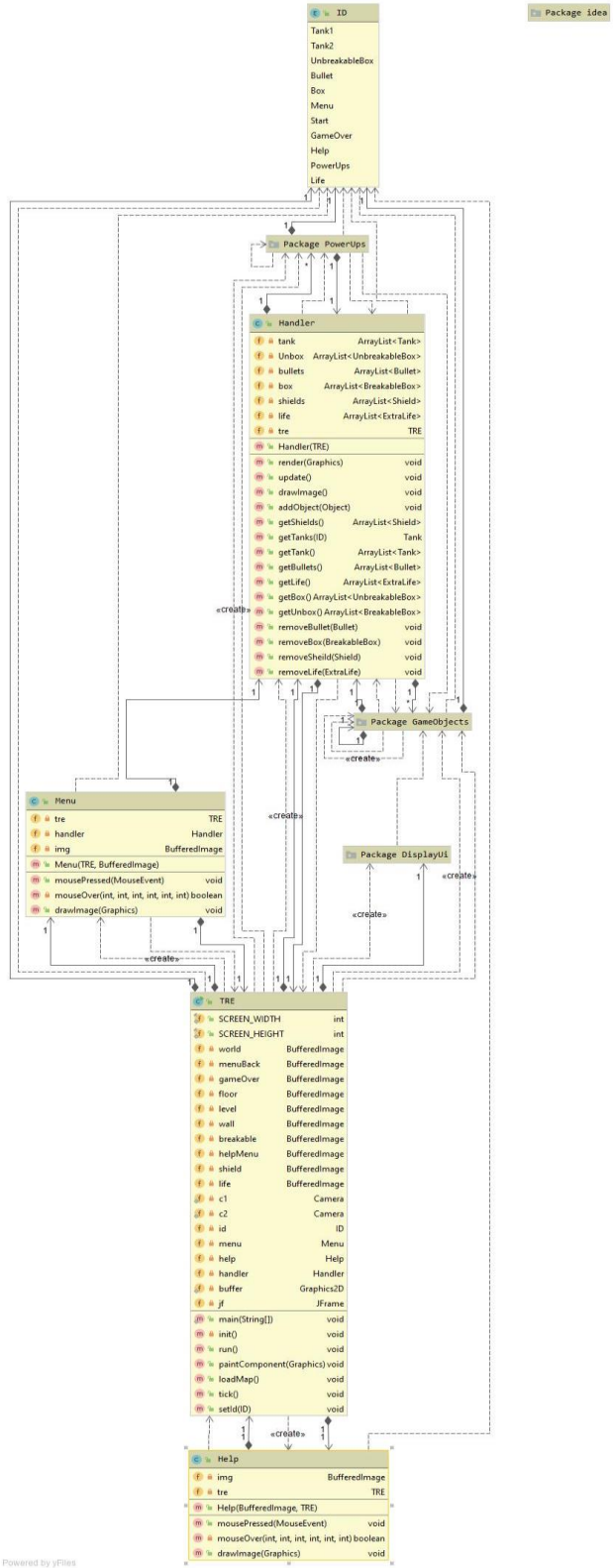
Assumptions were too many for this project, especially for me which was the first time doing a project this big. As I started I thought like most of my projects I would be able to design and go as I'm implementing the game but right at the beginning I was stuck. I realized I need to first make the connections between everything before coding it. My assumption about the data structure to hold my game objects came out to be a disaster due to having too many objects inside it, and I had to split up my data structure into different groups to reduce the updates for each class and not to update unnecessary classes to be able to make my game faster. The other assumption I remember was the about my power ups thinking of just counting them as a normal game object. But as I started implementing the power ups I realized it would be better to have the power ups as an abstract object same as all the game objects for a better OOP and it also helped me out in the moment the power up was picked up.

# 6 Implementation Discussion

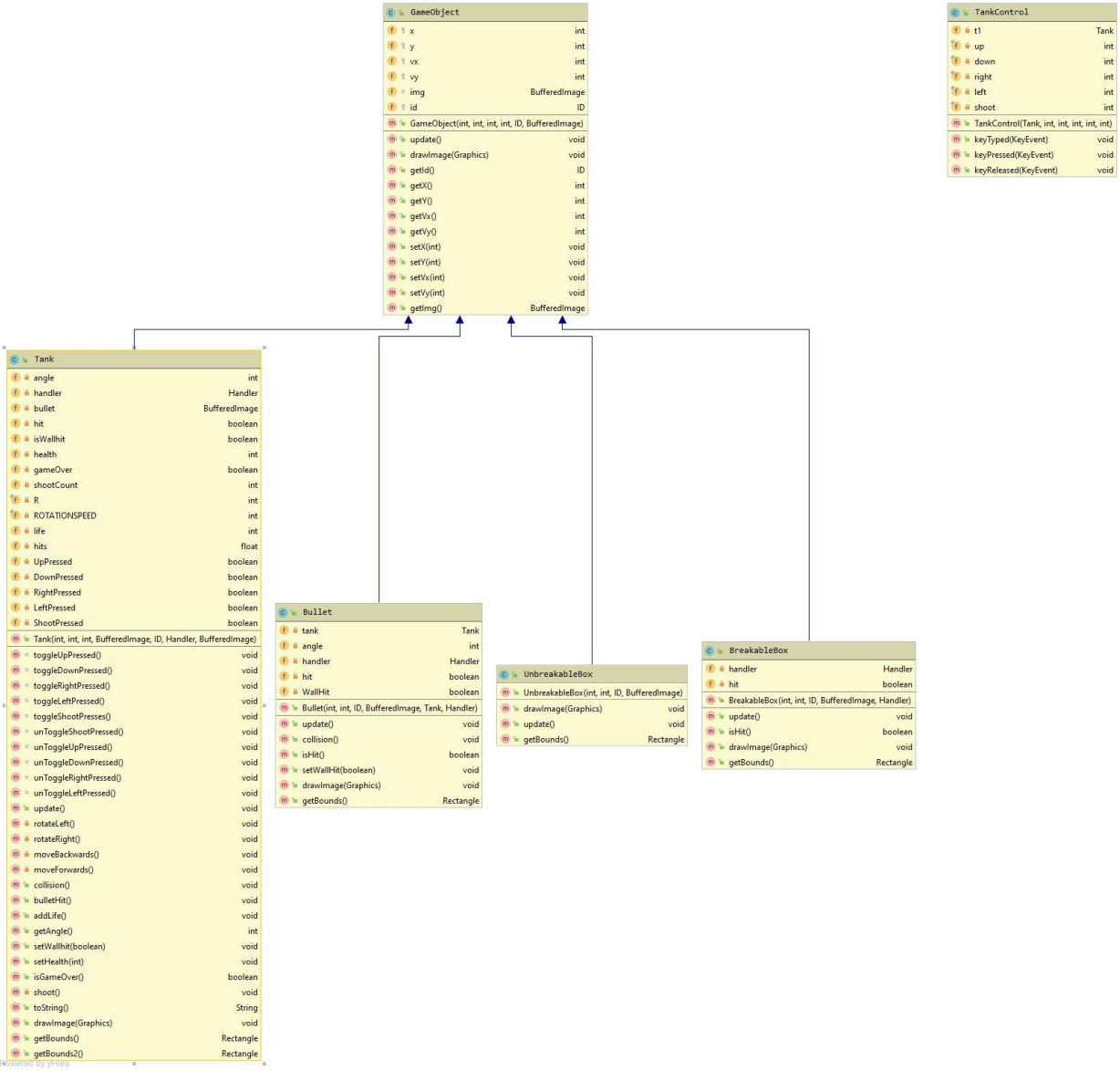
## 6.1 Class Diagram

I would import my diagram in this document but due to them being large and having too many contents I also added the .jpg file for each package diagram and the main package and each individual class and their relations with the document. (**Due to not being able to upload the pictures, I would recommend zooming in on the pictures to be able to read it better**) The class diagrams are big wasn't able to show them HD. Sorry for the Zoom.

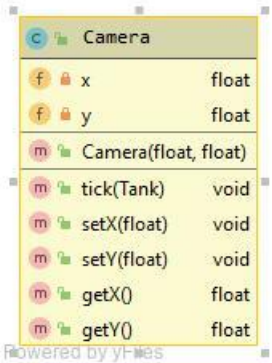
## 6.2 TankGame Package Diagram



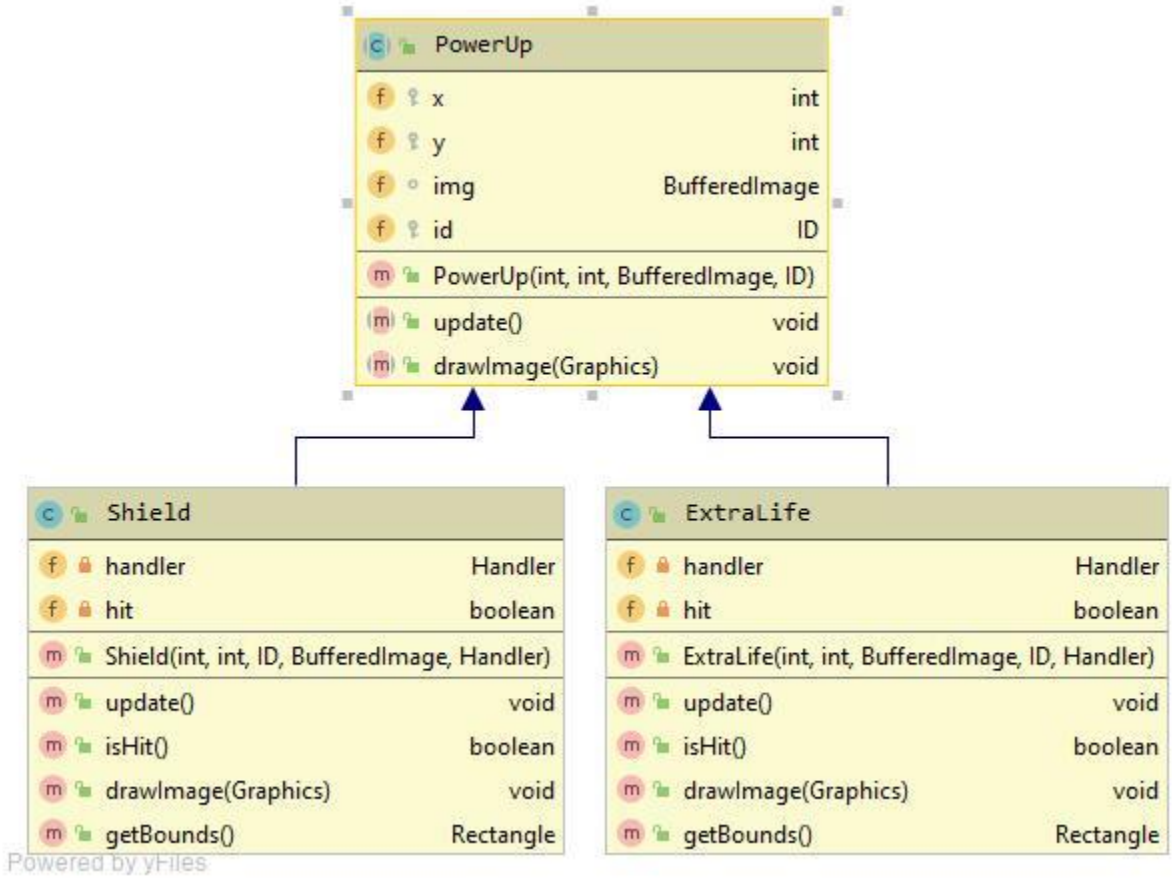
## 6.3 GameObject Package Diagram




## 6.4 Display Package Diagram



## 6.5 PowerUps Package Diagram




# 7 Class Description

## 7.1 Display Package Class Description

1. **Camera() Class**

   This class takes care of the fact each split screen is specific to each player. So what this class does is, It takes in the x and y of the specific Tank and by the help of that location it would generate a new x and y to be the new origin of the split screen so the tank is always in the middle and never gets lost.

   The class has a tick method which each frame it gets called and updates the location of the x and y by the formula inside it to generate the new origin point

   **The idea for this class was taken from the video source I mentioned in section 2.**

## 7.2 PowerUps Package Class Description

1. **PowerUp() {Abstract} Class**
   This abstract class is a basic class which takes in the x and y of the location of each power up which is respawned on the map and it also takes their Enum Id and image to be able to display it.
   This class also has 2 Abstract method:
   update() {Abstract} : Takes care of the collision if it happens and the changes each Power Up do
   drawImage(Graphics g) {Abstract} : Takes care of drawing the power up icon on map.
2. **ExtraLife() extends PowerUp Class**
   This class is for the power up Extra life. It would check inside its update function if the tank is collided with the object or not. If it did it would add an extra life to the total 3 life the tank has. And also the drawImage which appears the icon on the map.
   The class also has Boolean when it collides with the tank which with the help of the Boolean I delete the unnecessary objects from the data structure.
3. **Shield() extends PowerUp Class**
   This class is for extra health added to the player if needed. The object respawns on the map at the specific location and update checks for collision if true add more health to the player and also
   The class also has Boolean when it collides with the tank which with the help of the Boolean I delete the unnecessary objects from the data structure.

## 7.3 GameObjects Package Class Description

1. **GameObject() {Abstract} Class**
   This class is the main essence of this game. Everything inside the game that appears other than the power ups is a GameObject.
   Game Object consist of {Tank,Bullet,BreakableBox,UnbreakableBox}
   GameObject has also two abstract method:

   update() {Abstract} : Takes care of the collision if it happens and the changes each Tanks do

   drawImage(Graphics g) {Abstract} : Takes care of drawing the each Object icon on map.

   The Game Object class has 6 main variables which are important, such as x and y the position of the object, vx and vy which is the velocity for each object if it moves and Enum id to defer them from each other and lastly the image for each object.

2. **Tank() extends GameObject Class**
   The main class for the tank. This class takes care of everything related to the tank such as health, life, movement, direction, collision and if its dead. The class also takes in the handler class which contains all of the data inside the game
   The class consist of lot of Booleans which with the help of the class **TankControl** we turn them on and off and whenever they are on it would give command to move the tank. The tank moves only forward and backward with knowing which direction the head is facing.

a. **Update() Method**
   This method takes care of checking the Booleans to see if the tank has to move or not, and also calls the method collision to check for any collision if it happens. It also takes care of checking the life and health and if there is no more life to kill the game and gameOVer.
b. **Collision() Method**
   This class with the help of getters inside the handler class it would go through appropriate data structure to check if the boundaries of two tank and a bullet or Breakable or unbreakable wall is collided with the tank. This function also handles post collision to keep everything stable.
c. **bulletHit() Method**
   Gets called by the bullet class if it collides with the tank to add damage and reduce the health.
   **This class was taken from the Professor Anthony's TankRotationExample which was given to us.**

3. **TankControl() implements KeyListener**
   This class listens for keyboard input from the user. Also the class sets up the Controls for each Tank and also sets up a KeyListener for them to keep track of the input data.
   This class only turns on the Boolean inside the tank class for the tank to move and also turns of the Boolean when the button is released.
   **This class was taken from the Professor Anthony's TankRotationExample which was given to us.**
4. **BreakableBox() extends GameObject Class**
   This class is simple class that respawns a box on the map and the features of this box is it would destroyable. The class has access to the handler and also has Boolean for the moment the collision happens for removing the object from the game.
   The update function in this class loops through the tank and bullets available inside the handler data structure and checks for collision and if collision did happen it removes the object.
5. **UnbreakableBox() extends GameObject Class**
   This class doesn't do anything particular other than respawning it on the map and drawing it. UnbreakableBox has update class but it never gets called due to not needing it. The collision with this class is done inside the tank class due to being easier to handle the post collision for the tanks with the wall.
6. **Bullet() extends GameObject Class**
   Bullet would be the most complex one after the tank class in this package. My bullet class takes in Tank variable which would be the shooter and also cares about the angle of the tank at that moment for being able to draw the bullet in the correct direction. And lastly for being able to detect collision the bullet class has access to the handler object.
   The update in this class is for having the bullet move toward a specific location. So my update adds to the x and y the Sin and Cos of the angle of the tank to be able to calculate the vector to shoot toward.
   **Collision() Method :**
   This method takes care of looping through the breakable box and the enemy object to check for collision and if there is any collision a Boolean inside the bullet gets turned on which causes the

bullet to get removed after collision, and also takes care of the damage to each tank by calling the appropriate function.

## 7.4 TankGame (Main) Package Class Description

1. **TRE() Class (The Main Class Runs from here) Jframe**
   This class is where everything starts and ends. This class contains a the main function to run the program. The main function calls the init function and afterwards the run function which is the Game Loop.
   a. **Init() method**
      The init method starts up everything. Explaining from top to bottom. First init sets up the Jframe for the window with the correct size. Afterwards we make BufferedImage with the appropriate size which would be the size of the Map. My map is double the actual width and height of the screen. Afterwards we would read in any necessary image for the objects and background and help menu. After reading in the pictures, we start making new instance of the handler class, Camera class, Tank Control class and lastly we call the method loadMap which loads everything from a png file and sets up the map. We also call the constructor for menu and help class.
   b. **Run() method**
      This engine calls the handler update and drawImage Methods which refereshes all of the gameObject classes. Other than the update and drawImage I also check for appropriate objects to be removed and I remove it before the next frame comes.
      **This Method was taken from the Professor Anthony's TankRotationExample which was given to us.**

   c. **PaintComponent() method**
      This class first checks what state is the game in, if its menu show the menu if its game over finish the game or if it's the game state to play. This function which gets called by run, draws the map, background, tanks, blocks, and any other images in the game. This class which is override from the Jframe class gets called by calling reapint function from the run class.
      **This class was taken from the Professor Anthony's TankRotationExample which was given to us.**
   d. **LoadMap() method**
      Takes in a picture of png and reads it pixel by pixel and looks for different colors of red and makes a new instance of appropriate class and adds them to the handler data structure.
2. **Menu() Class extends MouseAdapter**
   This class extends the MouseAdapter casue we would be needing mouse input for the menu section. This class sets up the menu buttons and draws the appropite box around them, and if the mouse click is inside the bound the TRE Game State would change and it would draw the correct state if it's help or quit or play.
3. **Help() Class extends MouseAdapter**
   This class does the same as Menu with the difference of cause I needed to return from my help section to the normal state I needed to implement it in a different class.
4. **Handler() Class [Heart of the Game]**
   This class is where all the update and drawImage abstract method of each game Object gets called. The handler class has 6 different data structures inside it.

1. Tank ArrayList
2. UnbreakableBox ArrayList
3. BreakableBox ArrayList
4. Bullets ArrayList
5. Box ArrayList
6. Sheilds ArrayList
7. Life ArrayList

This class has one addObject Method which takes in any type of Object and if it's the instance of the correct objects we have we add it to the data structure. Also all of the data structures have getter for the time we need to loop through them at different time, and also wrote remove method for Bullet Box and Shield and Life

# 8 Project Reflection

I learned a lot as I did this project this semester. I started by panicking about the size of the project and not knowing anything but it was like rollercoaster scary at first but then everything fell into the right place. This project helped me to understand the idea of parent and child and how important this relation is. Data Structure in the program make huge difference if it's not accessed properly or not the right type. My game started at first pretty slow but as I learned and went through I changed couple of OOP design in my program and my data structure I split them up and this caused my game from running annoyingly slow to smooth fast game. That was when I felt the true power of OOP design and data structure.

# 9 Project Conclusion/Results

The result was amazing tank game which I was able to write from scratch. The project was long and bigger size then I'm use to usually. This project thought me time management in project is important. How much Class diagram is important before starting to code and it's for better good to design and connect everything before coding which made be excited about the idea of learning how to design my project before starting. Overall I'm grateful for this project and how it made complete different programmer then I came to this class. Everything about this project was amazing