Aryanna Brown
CSC 413.01 SU20
https://github.com/csc413-01-SU2020/csc413-tankgame-aryannayazmin.git

1. Introduction
    a. In this project I implemented a Tank Wars game to practice good OOP. This includes creating classes that are useful, no hard coding, and making sure that each object can be easily updated without breaking the project.
    b. The game is a split screen two-player game involving two tanks battling. There is also a mini map that shows the whole map and movements of the tanks as the game goes on. There are breakable walls and unbreakable walls your bullet can collide with. Each tank has three lives and one health bar starting at 100. Each hit you take from a bullet it -5 off your health.
2. Development Environment
    a. Java 11
    b. IntelliJ Ultimate 2020
    c. All of my resources are from ilearn
3. How to build or import game in IntelliJ
    a. Clone repository link and open a terminal and type git clone “link”. Import project into IntelliJ and click the play.
    b. To build the jar I used Maven and fixed the configurations to compile through maven each time it ran to be sure the jar was updated.
    c. To run the game using JAR, use the command line to cd into the project folder, then cd jar , then run java -jar csc413-tankgame-aryannayazmin-1.0-SNAPSHOT.jar
4. How to run game
    a. After importing game, click on Jar file then press play.
    Player 1 Controls (Left Screen):
    Up arrow = forward down arrow = backward left arrow = rotate left
    right arrow = rotate right enter = shoot
    Player 2 Controls (Right Screen):
    W = forward S = backward A = rotate left D = rotate right Space = shoot

    The game is a split screen two-player game involving two tanks battling. There is also a mini map that shows the whole map and movements of the tanks as the game goes on. There are breakable walls and unbreakable walls your bullet can collide with. Each tank has three lives and one health bar starting at 100. Each hit you take from a bullet it -5 off your health. Once a player loses all their lives, the game will exit automatically. There are three powerups that can be obtained when playing. Health: Brings back your health bar to 100. Only works if health is less than 100. Speed Up: Increases tank speed. Life Up: Increase your lives by +1.
5. Assumptions Made

    The assumptions made in the game were that people may know the basic rules of a game, staying alive. Another assumption made is that each player has a keyboard with arrows.
6. Tank Game Class Diagram

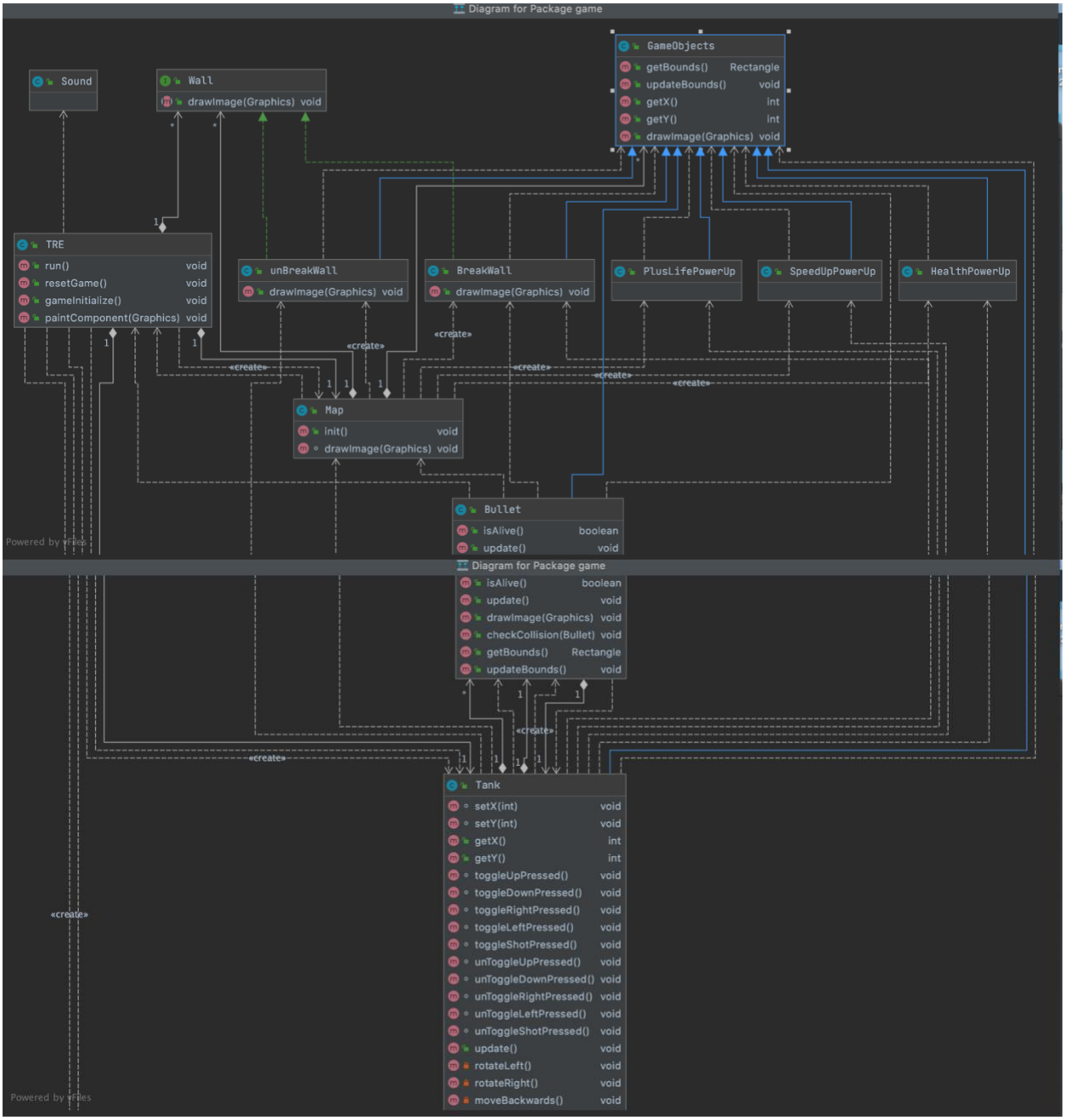
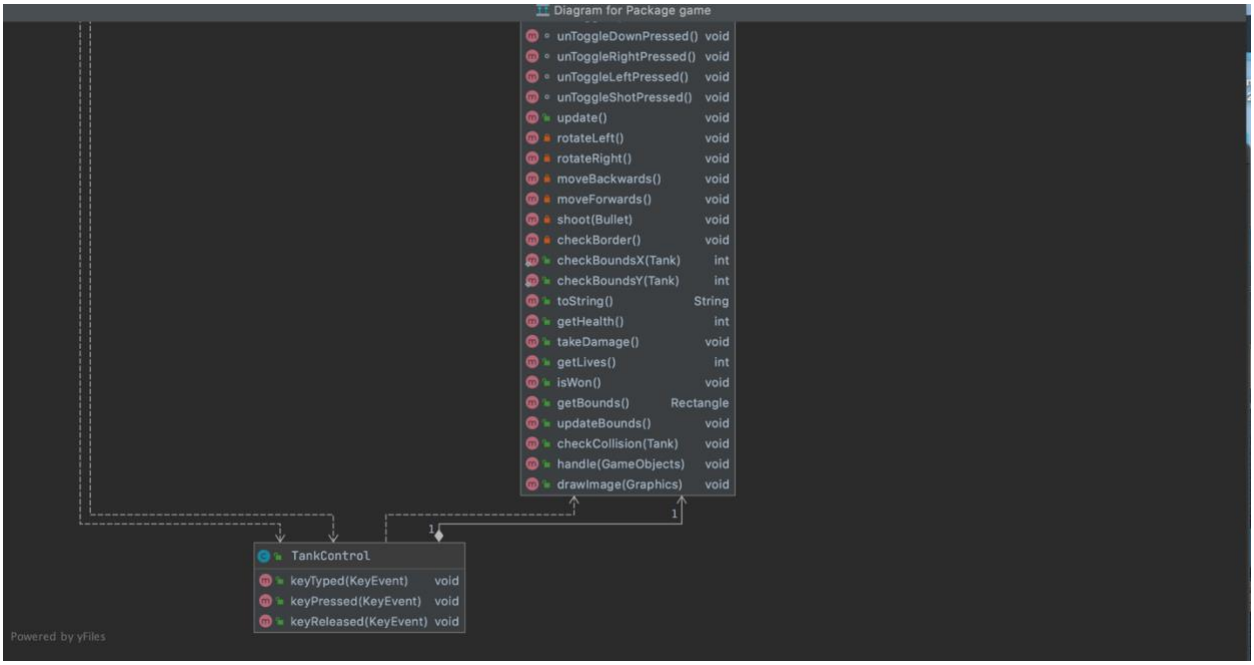


Diagram for Package game
GameObjects
getBounds() Rectangle
updateBounds() void
getX() int
getY() int
drawImage(Graphics) void
Sound
Wall
drawImage(Graphics) void
TRE
run() void
resetGame() void
gameInitialize() void
paintComponent(Graphics) void
unBreakWall
drawImage(Graphics) void
BreakWall
drawImage(Graphics) void
PlusLifePowerUp
SpeedUpPowerUp
HealthPowerUp
Map
init() void
drawImage(Graphics) void
Bullet
isAlive() boolean
update() void
«create»
Powered by yFiles
Diagram for Package game
isAlive() boolean
update() void
drawImage(Graphics) void
checkCollision(Bullet) void
getBounds() Rectangle
updateBounds() void
Tank
setX(int) void
setY(int) void
getX() int
getY() int
toggleUpPressed() void
toggleDownPressed() void
toggleRightPressed() void
toggleLeftPressed() void
toggleShotPressed() void
unToggleUpPressed() void
unToggleDownPressed() void
unToggleRightPressed() void
unToggleLeftPressed() void
unToggleShotPressed() void
update() void
rotateLeft() void
rotateRight() void
moveBackwards() void
Powered by yFiles



7. Class Descriptions

BreakWall and unBreakWall both are implemented and added to the gameObjects array. Bullet has a method that checks if the tank is alive or still has lives. It also has a method that will check if it collided with anything such as a wall or a power up or a tank. The Bullet class will also update objects that are hit with a bullet and checks the bounds of the bullet to be sure it doesn't go off the map and crash the game. The class GameObjects gets the bounds of each object and helps draw each of them while instantiating them as objects. Health Power Up class adds the object to GameObjects. The Map class initializes the map for the whole game that includes the background, walls, and powerups. PluslifePowerUp class adds the object to GameObjects to later be implemented in the map. The Sound class holds the sound for the game. The SpeedUpPowerUp class adds the object to GameObjects to later be implemented in the map. The Tank class sets each of the buttons pressed and the actions associated. It also updates the tanks movement

and updates the shots being fired. The methods for each power up are also implemented in this class, as well as the amount of lives and who is winning the game. The tank class also checks the collisions and handles the shooting of the walls. The Tank control class just implements what key is pressed and released. The TRE class runs the game, resets the game, and initializes it. It also paints the lives, health bars, and tanks in the game. The Wall interface helps paint the walls.

8. Self Reflection on Development

   The development process of the project was hard to go through in navigating good OOP. The stress of not fully knowing whether or not something was acceptable made implementing new things hard. It was very fun to see each new thing be implemented into the game as I progressed. Making sure each class had an important role to play and implemented it well.

9. Project Conclusion

   Overall the project was very fun but difficult. The weight of one project on my grade made implementing the game a bit more stressful than need be. The project did teach me to pay attention to detail when coding and not to have unnecessary things in places that may not matter in the overall game.