

CSC 413 Project 3 Documentation

Summer 2024

Devon Huang

916940666

csc413-01

<https://github.com/csc413-SFSU-SU2024/csc413-tankgame-Novedh>

Table of Contents

1. Introduction.....	3
a. Project Overview (focus of term project).....	3
b. Introduction of the Tank game (general idea).....	3
2. Development environment.....	3
a. Version of Java Used.....	3
b. IDE Used.....	3
c. Any special libraries used or special resources and where you got them from.....	3
3. How to build or import your game in the IDE you used.....	4
a.....	4
Go to https://github.com/csc413-SFSU-SU2024/csc413-tankgame-Novedh	4
b. List what Commands that were ran when building the JAR. Or Steps taken to build jar.....	6
c. List commands needed to run the built jar.....	10
4. How to run your game. As well as the rules and controls of the game.....	12
5. Assumptions Made when designing and implementing your game.....	12
6. Tank Game Class Diagram.....	12
7. Class Descriptions of classes implemented in the Tank Game.....	12
a. No need to over explain but simply state the purpose of the class.....	12
8. Self-reflection on Development process during the term project.....	12
9. Project Conclusion.....	12

1. Introduction

a. Project Overview (focus of term project)

The purpose of this project was to write a 2d game written in java and to practice object oriented programming.

b. Introduction of the Tank game (general idea)

The game had certain requirements such as having two players control a tank by shooting bullets until a tanks health depletes. It must have a split screen and a mini map. 3 lives per tank and a health bar. For the power ups I used a speed boost, heal pickup, and shield. For the animations I added tracks when the tank moves that fades, shooting animation, and bullet colliding animations. For sounds i have a running tank sound that gets louder when the tank is moving and quieter when the tank is idle, i also have a shooting sound and a colliding sound.

2. Development environment.

a. Version of Java Used

Java version 22.0.2

b. IDE Used

IntelliJ IDEA 2021.2.3 (Ultimate Edition)

c. Any special libraries used or special resources and where you got them from.

I used a couple assets that I got online, I used a bigger bullet and a bubble for when the tank is shielded and hearts for keeping track of lives. Then I took the menu title and changed the colors and tests to show which tank won for the end title.

3. How to build or import your game in the IDE you used.

a.

To Import :

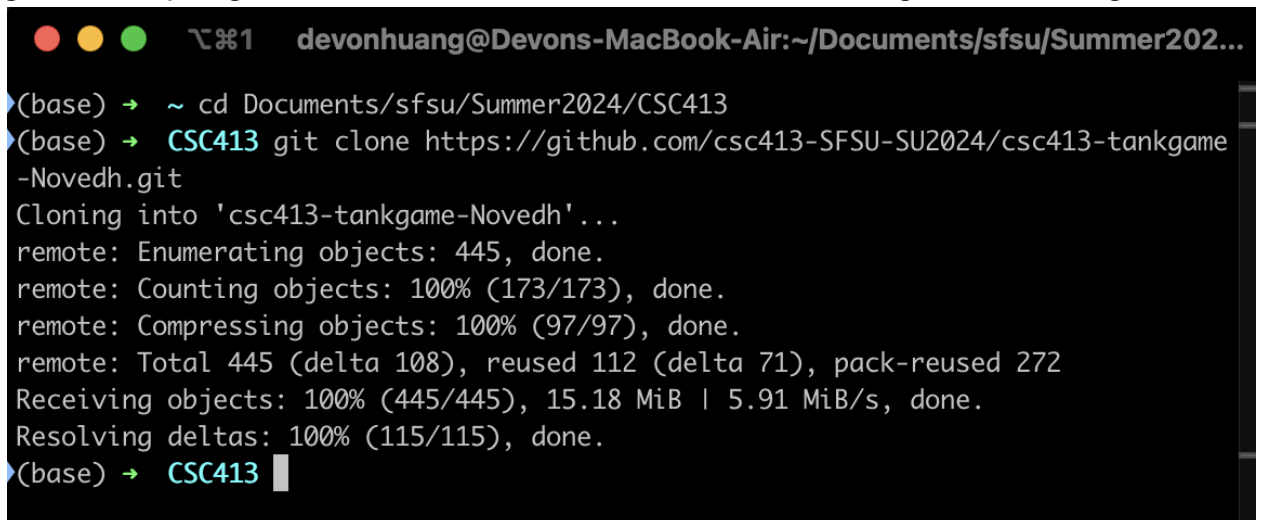
Go to <https://github.com/csc413-SFSU-SU2024/csc413-tankgame-Novedh>

1. click on code and download the zip
2. unpackage the zip and move the folder to a good place
3. Open your IDE (IntelliJ or VSCode)
4. Open project on the project folder
5. Project has been imported!

Or from command line:

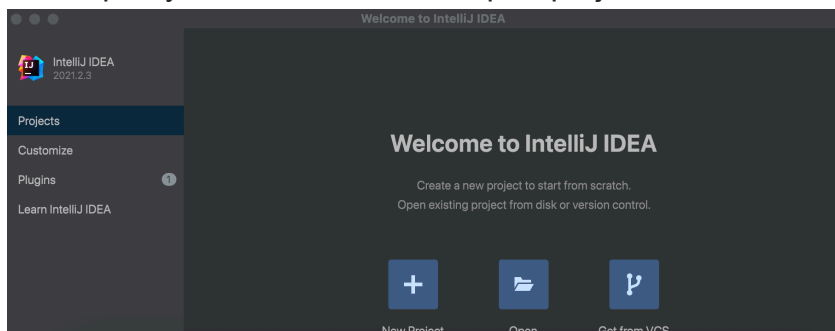
1. open terminal
2. cd to a good place, such as documents
3. run

`git clone https://github.com/csc413-SFSU-SU2024/csc413-tankgame-Novedh.git`

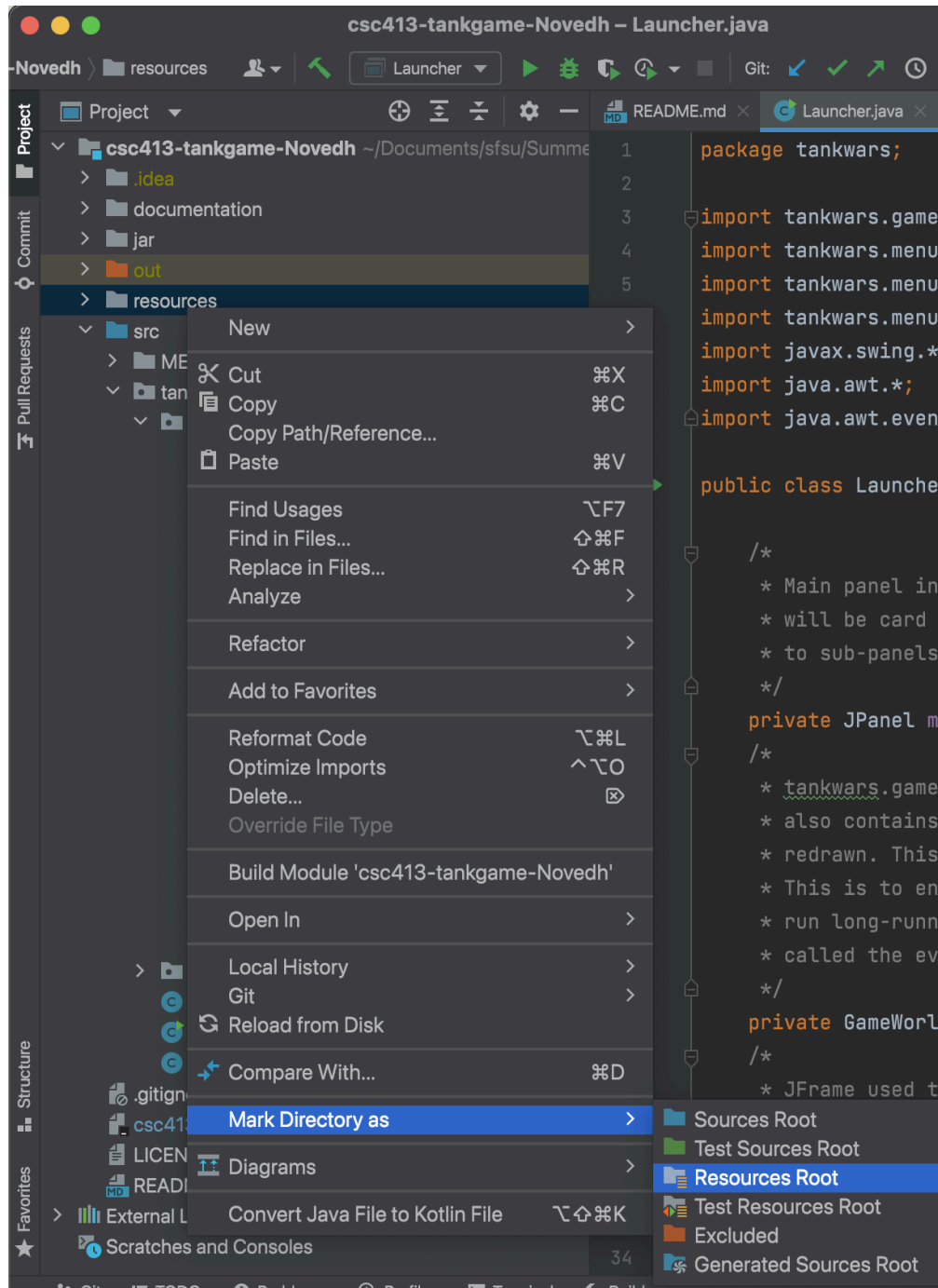
A terminal window on a Mac with a dark background. The title bar shows the user 'devonhuang' on a 'Devons-MacBook-Air' in the directory '~/Documents/sfsu/Summer202...'. The prompt is '(base) →'. The user enters 'cd Documents/sfsu/Summer2024/CSC413' and then 'CSC413 git clone https://github.com/csc413-SFSU-SU2024/csc413-tankgame-Novedh.git'. The output shows the cloning process: 'Cloning into 'csc413-tankgame-Novedh'...', 'remote: Enumerating objects: 445, done.', 'remote: Counting objects: 100% (173/173), done.', 'remote: Compressing objects: 100% (97/97), done.', 'remote: Total 445 (delta 108), reused 112 (delta 71), pack-reused 272', 'Receiving objects: 100% (445/445), 15.18 MiB | 5.91 MiB/s, done.', 'Resolving deltas: 100% (115/115), done.', and finally '(base) → CSC413' with a cursor.

```
devonhuang@Devons-MacBook-Air:~/Documents/sfsu/Summer202...  
(base) → ~ cd Documents/sfsu/Summer2024/CSC413  
(base) → CSC413 git clone https://github.com/csc413-SFSU-SU2024/csc413-tankgame-Novedh.git  
Cloning into 'csc413-tankgame-Novedh'...  
remote: Enumerating objects: 445, done.  
remote: Counting objects: 100% (173/173), done.  
remote: Compressing objects: 100% (97/97), done.  
remote: Total 445 (delta 108), reused 112 (delta 71), pack-reused 272  
Receiving objects: 100% (445/445), 15.18 MiB | 5.91 MiB/s, done.  
Resolving deltas: 100% (115/115), done.  
(base) → CSC413
```

4. Then open your IDE and select import project and select the project folder



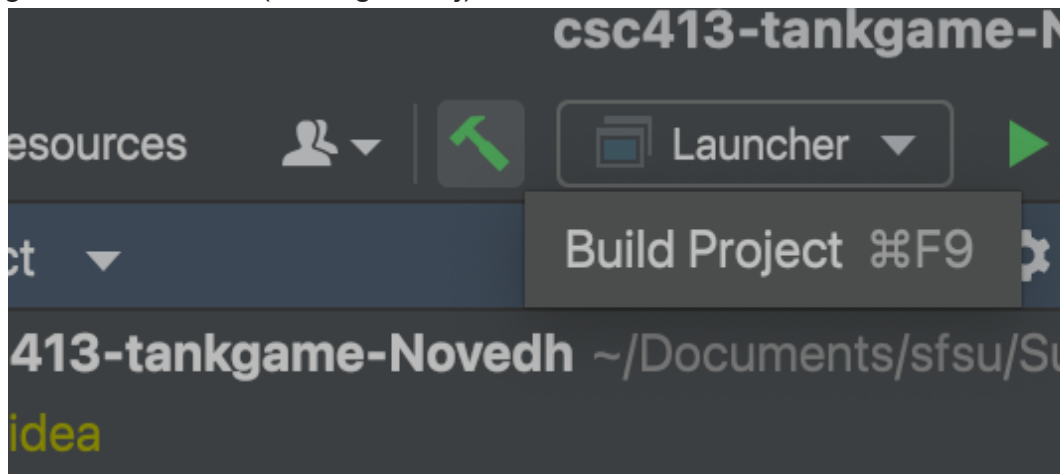
5. After set resources as root resource folder, might have to invalidate caches as well and set src as source root.



- 6.

To Build :

1. Have java installed
2. Open IDE
3. go to the Hammer(if using IntelliJ) and click on it to Build.

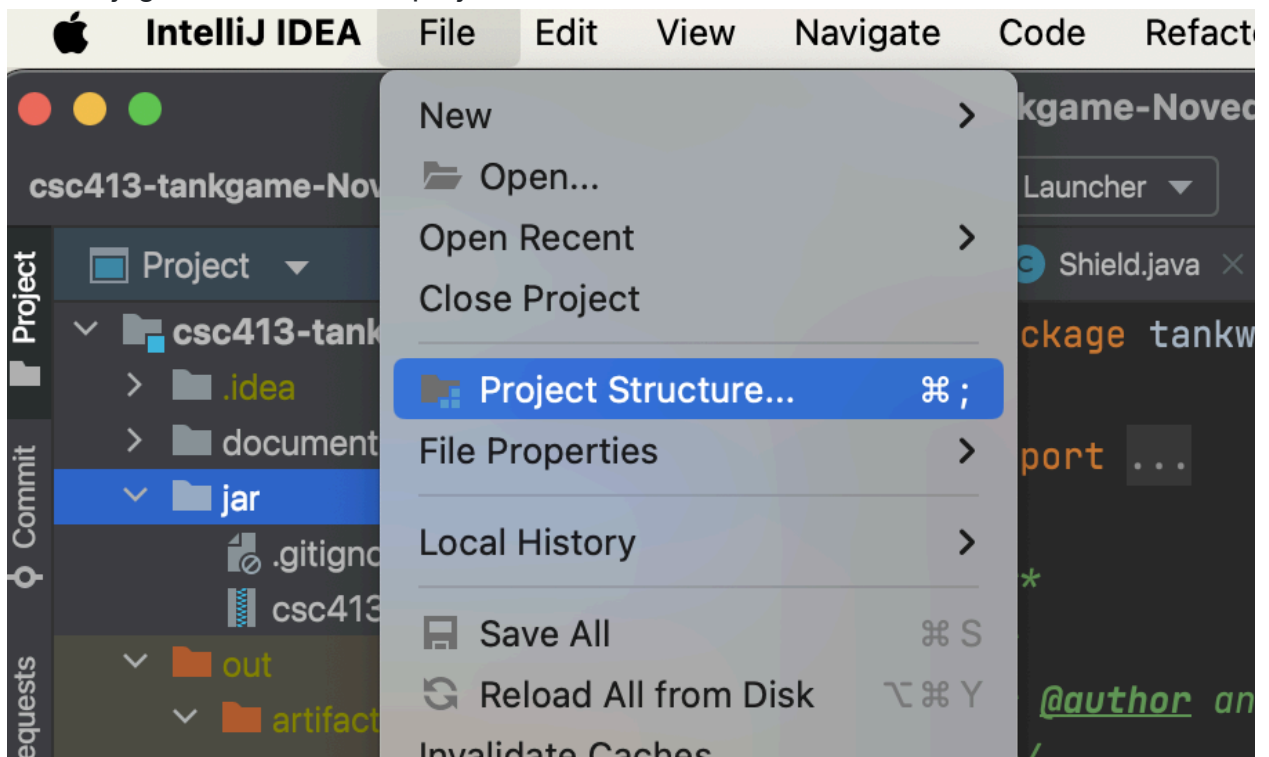


b. List what Commands that were ran when building the JAR.

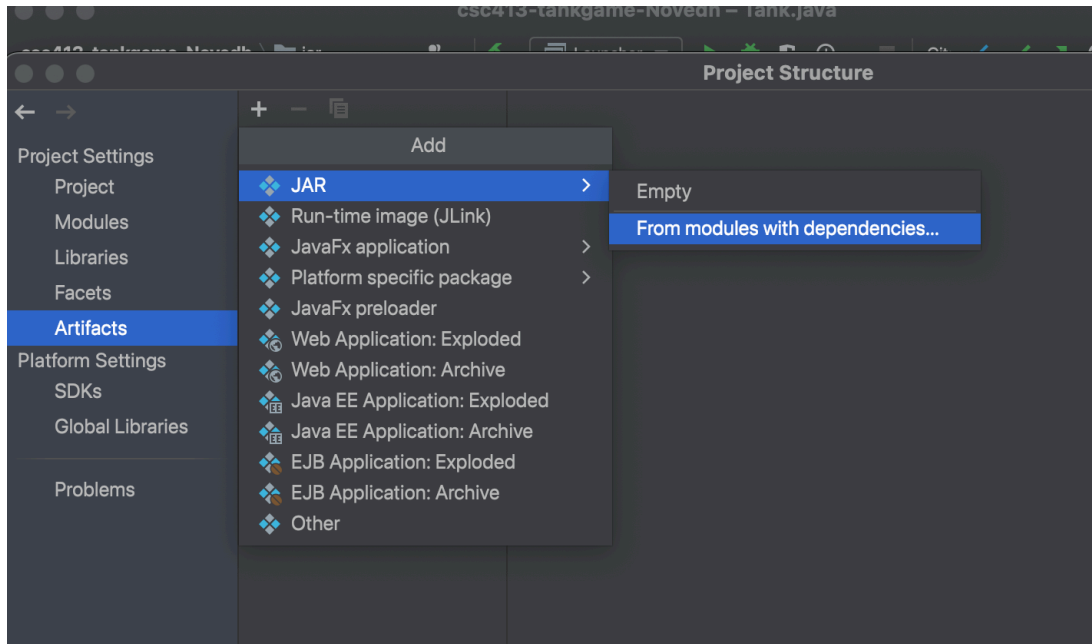
Or Steps taken to build jar.

To Build the jar :

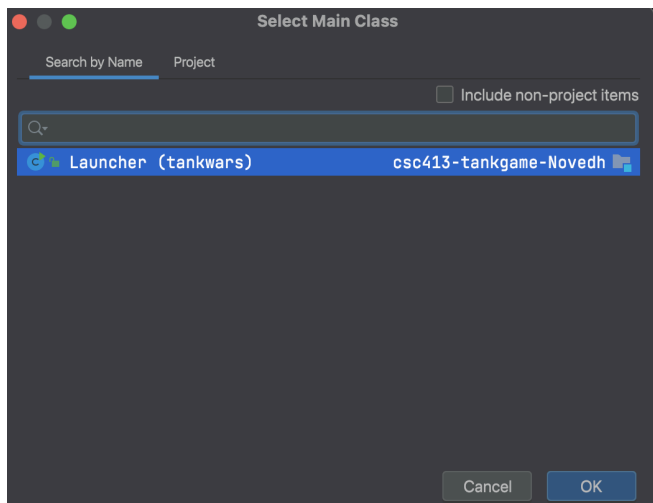
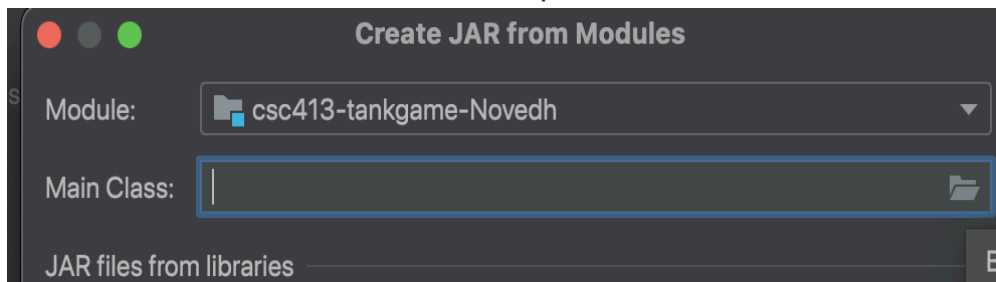
1. In IntelliJ, go to File and then project structure



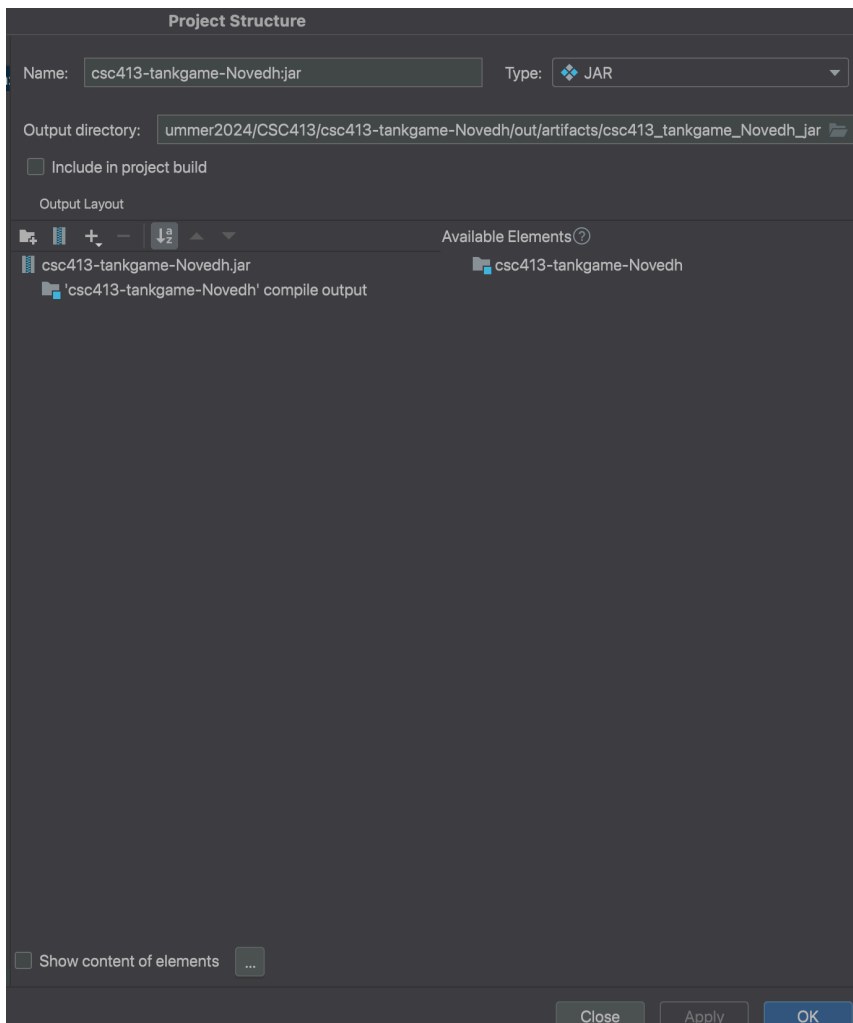
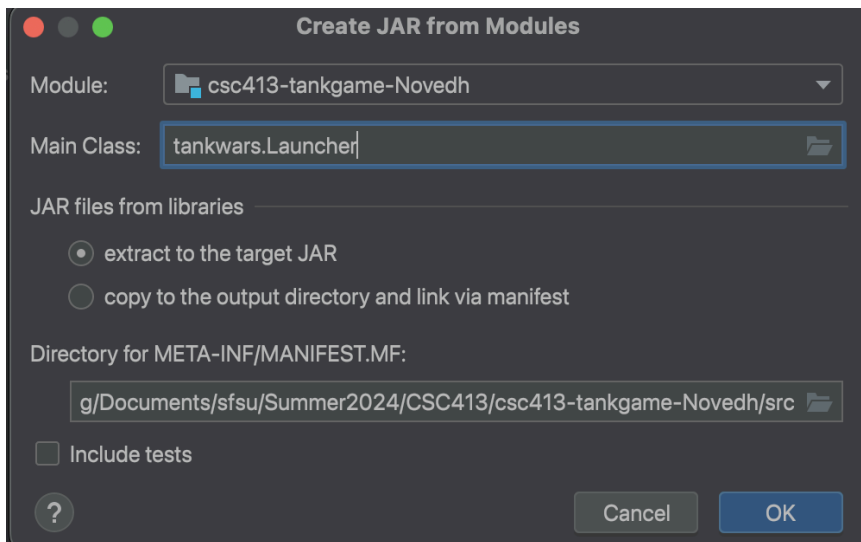
2. then on the left side, click Artifacts and the plus sign on the next tab and under Jar, click from Modules



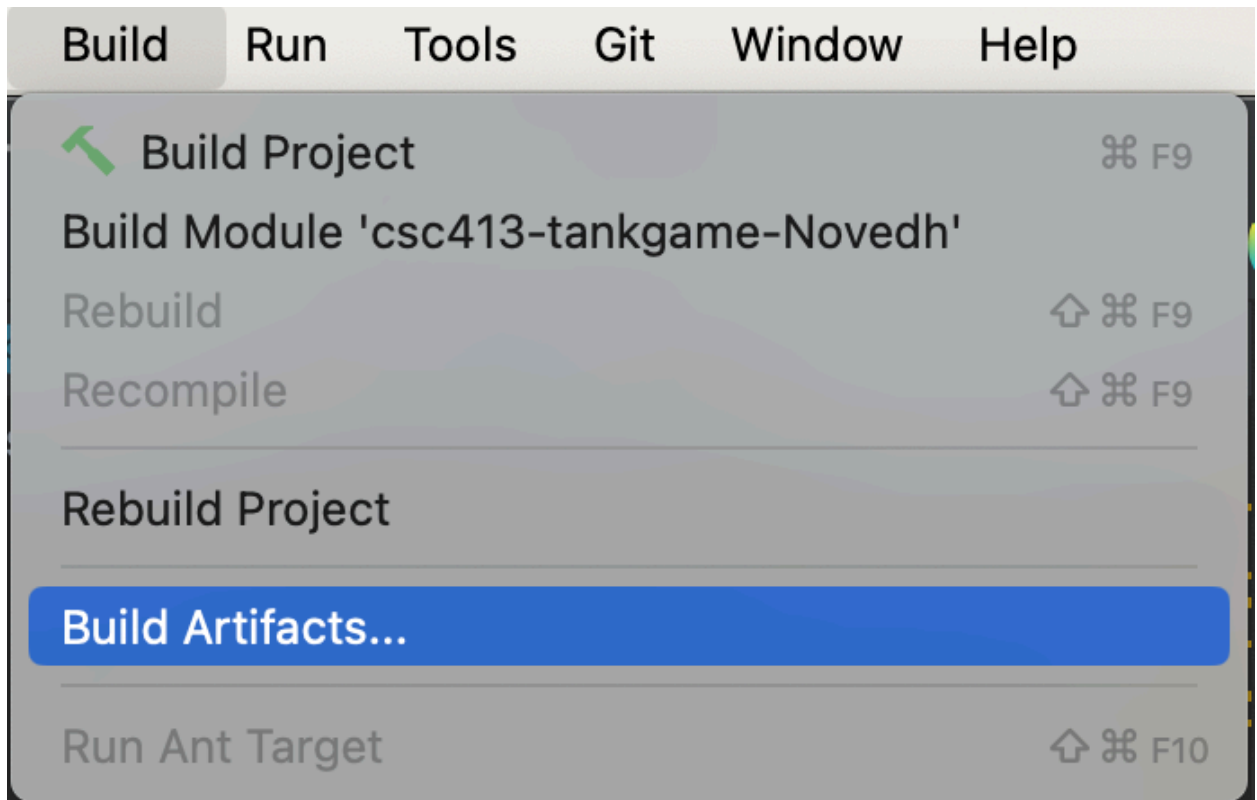
3. then select main class, Launcher and press OK



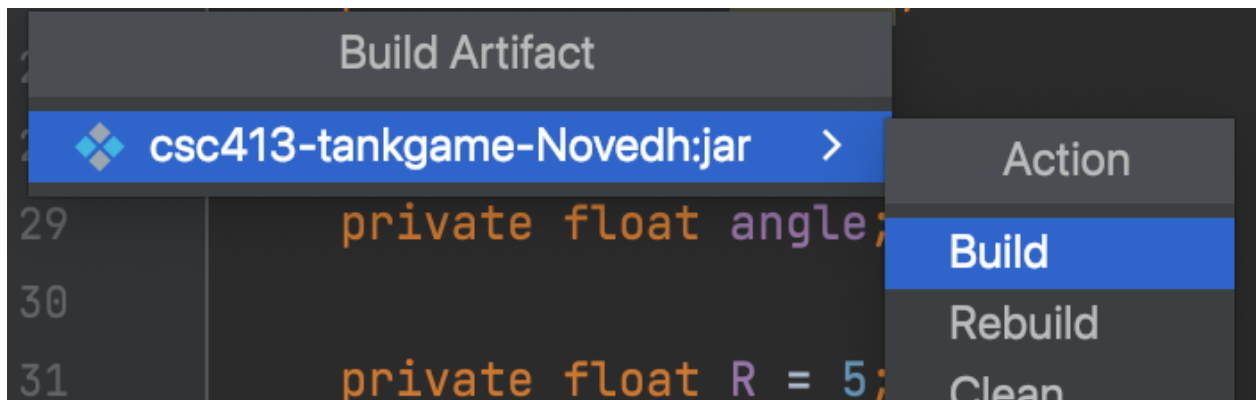
4. And press OK, Apply and OK again to create jar



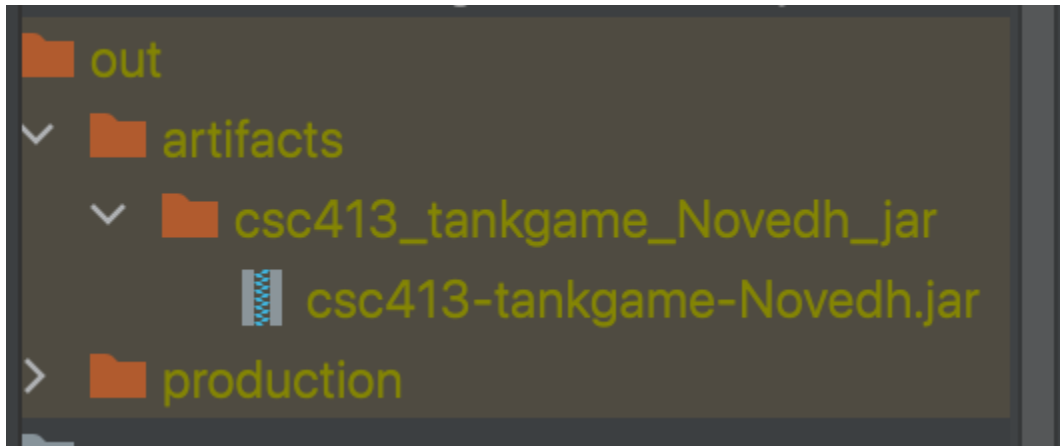
5. Then go to build Tab and Build artifacts



6. And click build



7. It will now be in the out/artifacts/csc413-tankgame-Novedh_jar file



c. List commands needed to run the built jar

1. Open terminal
2. then cd to the project location
3. then type java -jar ./jar/csc413-tankgame-Novedh.jar

```
➤(base) → csc413-tankgame-Novedh git:(main) java -jar ./jar/csc413-tankgame-Novedh.jar
```

4. then game will start



4. How to run your game. As well as the rules and controls of the game.

Game will run when you click start after running the jar/ through the play button on intelliJ

The rule is to reduce the enemy tank's health to 0 to take a life away, after 3 lives the game is over and the winning tank gets an end screen. From there you can exit or replay.

Controls to play your Game:

	Player 1	Player 2
Forward	W	Up
Backwards	S	Down
Rotate Left	A	Left
Rotate Right	D	Right
Shoot	Space	Enter/Return

5. Assumptions Made when designing and implementing your game.

The assumptions made was that the game starts both tanks with 3 lives and the game ends after a tank has no more lives. There will be two players and each will be controlled independently. The tank will shoot a projectile that will do damage to another tank or break breakable walls. Tanks will also have three power ups that they can use.

6. Tank Game Class Diagram

Diagram from milestone 1:

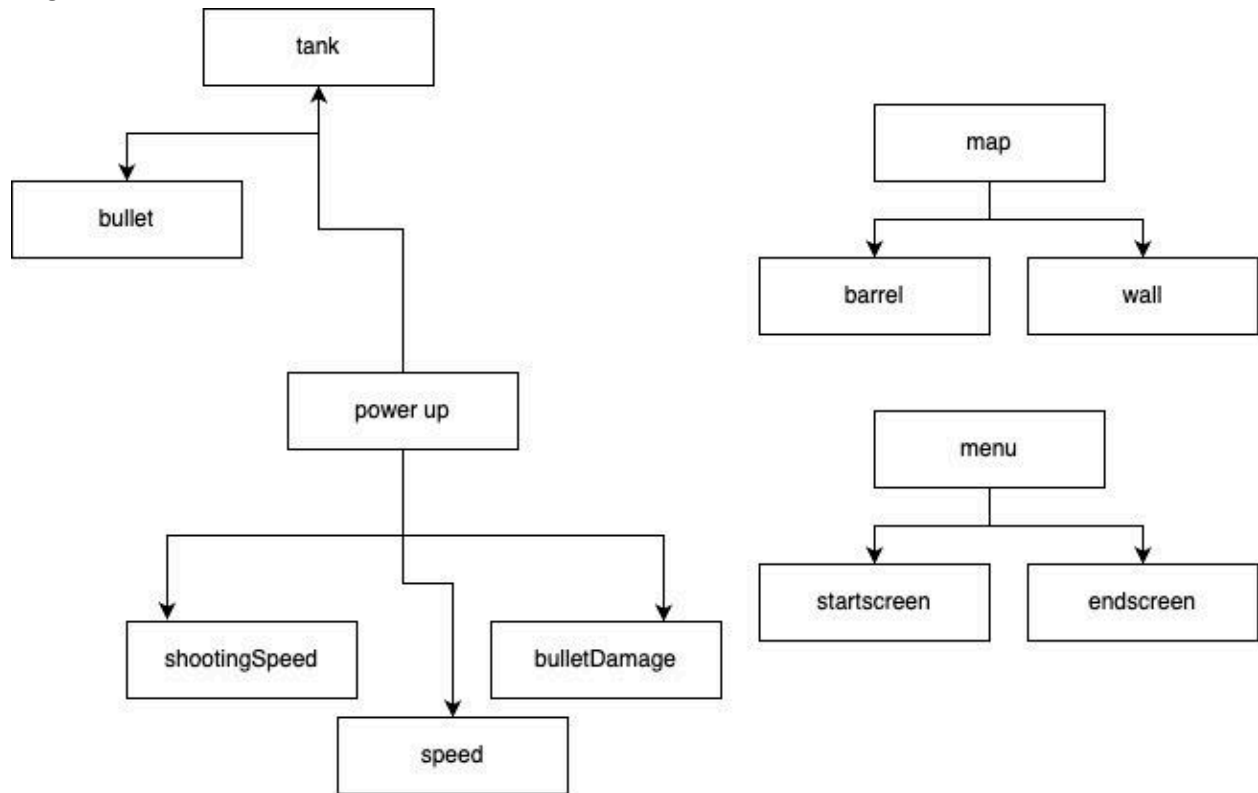
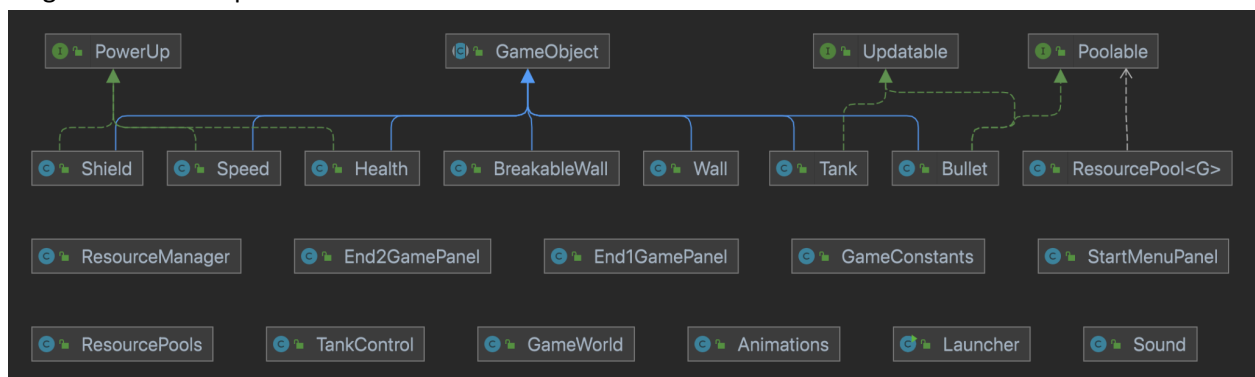


Diagram when completed:



7. Class Descriptions of classes implemented in the Tank Game

1. Animations: manages frames of image to create a animation
2. BreakableWall: a wall that can be broken, holds onto its position and if its has been hit
3. Bullet: a bullet that handles collisions and based on what it hits and draws a bullet
4. GameObject: The superclass that all objects inherit from
5. GameWorld: in charge of the games map, hold objects such as the walls and tanks and draws them
6. Health: a powerup that when collided heals the tank
7. Poolable: a interface of a poolable item that can reset and be initialized
8. PowerUp: a interface for the powerups
9. ResourcePool: manages the poolable objects adding objects in a pool and removes them
10. ResourcePools: manages multiple ResourcePool instances
11. Shield:a powerup that when collided shields the tank
12. Sound: in charge of audio, allows playing, stopping and looping audio
13. Speed: powerup that when collided speeds up the tank
14. Tank: a tank object hold onto the tanks positions and allows it to move
15. TankControl: sets up the keys to move the tank
16. Updatable: a interface for objects that can be updated
17. Wall: a wall that can't be broken.
18. End1GamePanel: the end panel for when player 1 wins
19. End2GamePanel: the end panel for when player 2 wins
20. StartMenuPanel: the start screen that allows players to start the game
21. GameConstants: holds onto constants such as spawn points, screen size and map size.
22. Launcher: the launcher for the game, sets the panel for the game to be played and loads assets.
23. ResouceManager: in charge of loading sounds, animations, and sprites into the game.

8. Self-reflection on Development process during the term project

In the beginning I looked at how much work needed to be done for a complete project and wondered how I would be able to complete such a task, but with the guidance and tips on what to do by the professor I was able to turn-in a functional game. I liked that there was freedom on how each person would implement certain things like animations and sounds because I had a lot of fun trying to figure out how I could make the tanks engine louder while it was moving and how I could make the tank take no damage while the shield was on. This project was fun because it was interactable, it was not just a calculator it was something that I could have fun playing and mess around with. I was able to see how much it changed starting from a skeleton code to an actual playable game, It shows how adding small features everyday can eventually create a big program with many functionalities.

9. Project Conclusion.

So the game works, I think it has all the requirements. The only thing is that it could have been better designed when it came to OOP practices. I'm happy with what it is, with more time and experience the game could be more.