CSC 413 Term Project Documentation

Summer 2019

Ye Lu

918128502

CSC413.01

~~https://github.com/csc413-01-summer2019/csc413-p2-monoluk.git~~

Table of Contents

[1 Introduction 3](#_Toc15985425)

[1.1 Project Overview 3](#_Toc15985426)

[1.2 Introduction of the Tank game (general idea) 3](#_Toc15985427)

[2 Development Environment 3](#_Toc15985428)

[3 How to Build/Import your Project 3](#_Toc15985429)

[4 How to Run the Game. Rules and controls of the game. 3](#_Toc15985430)

[5 Assumption Made 3](#_Toc15985431)

[6 Tank Game Class Diagram 4](#_Toc15985432)

[7 Tank Game Class Descriptions 5](#_Toc15985433)

[8 Self-Reflection on the project 5](#_Toc15985434)

[9 Project Conclusion 5](#_Toc15985435)

# Introduction

## Project Overview

This project is a 2D game written in Java. It is implemented using Object Oriented programming as main concept. JFrame and JPanel are used for the Graphic Users Interface (GUI) of the game.

## Introduction of the Tank game (general idea)

The game is about a Tank shooting war. It consists of 2 players (controlled by human), and a random number of enemies (controlled by computer). The main goal of the game is for the 2 players to shoot and kill each other. The survivor will win the game. The enemies were added to shoot at the players to add difficulties to the game. Special powerup(s) is available to increase the players’ chance of winning.

# Development Environment

IDE: IntelliJ

JDK: 12.0.0

Other Libraries: N/A

# How to Build/Import your Project

1. Open inelliJ
2. Select Import Project
3. Select the “csc413-tankgame-monoluk” folders as the source root of your project
4. Keep the “Create project from existing resources” radio button selected
5. Leave All default fields as they are
6. In “Please select project SDK” screen, choose a pre-installed JDK from your machine
7. You should be able to see the Project browser if everything is done correctly

# How to Run the Game. Rules and controls of the game.

## How to run the game

**-Run Game from main file (Launcher.java)**

(In intelliJ), right click the Launcher.java file (on the left side), and select “Run Launcher.main()” to run program(game).

**-Run Game from TankWar.jar**

Go to folder “csc413-tankgame-monoluk”, then go to “jar”, right click TankWar.jar file, and select “Jar Launcher”.

**-Run TankWar.jar from Comment Line**

$ java -jar tankWar.jar

## Rules of the game

Each player will have 3 lives in total. Each live contains 3 Health points. They will be placed in a battle field, they will chance and shoot at each other. If a bullet hits a player, this player will lose 1 health point. If a player lose 3 health points, one life will be deducted from this player. If one player lose all 3 lives, the other player wins the game.

**Enemies**

A random number of enemies tanks(wave) are added to the game. One enemy will contain only one health point and one life. They will shoot at the player randomly, at a random rate of firing. If one wave of enemies is cleared by the players, a next wave containing one more enemies then the last wave will be generated. This process will continue until a winner is produced.

**Powerup**

If an enemy is shot by the player, there is a 50% chance that a powerup will appear at the spot where the enemy explodes. Either player can pickup the powerup. After a player pickup a powerup, all the enemies on the screen will only shoot at the other player for 5 seconds. A powerup will disappear after 10 seconds if no player picks it up.

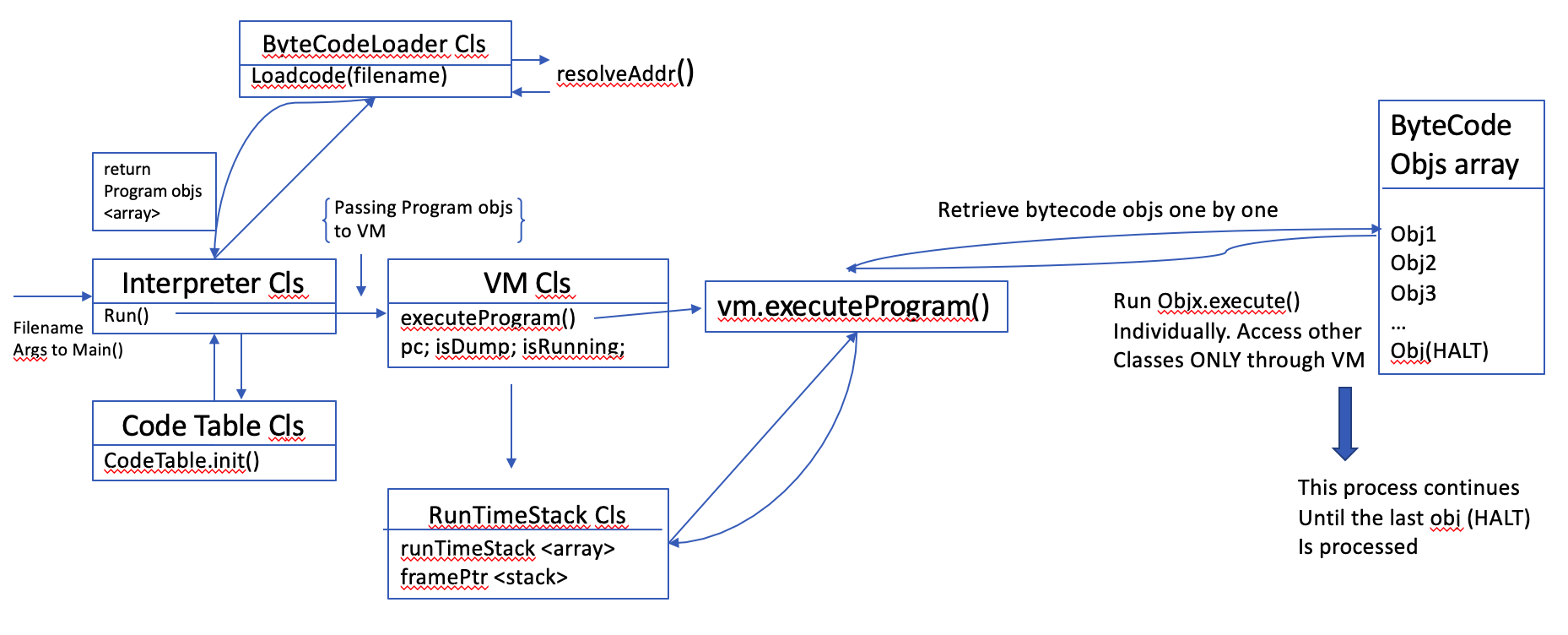
**Walls**

There are two types of wall, breakable and unbreakable. Breakable wall can be destroyed by bullets. Unbreakable wall cannot be destroyed. If a player run into a wall, breakable or unbreakable, it will be stopped by the wall. If an enemy run into the same wall, it will be bounded back to it’s original direction.

# Assumption Made

The computer that runs this game have JVM or JDK installed. Also the resolution of the same computer will be greater or equal to 1530\*900 pixels.

# Tank Game Class Diagram



~~The program starts from the interpreter class, by acquiring the filename from the argument passed to Main();~~

~~The interpreter initialize a codeTable object in the codeTable class.~~

~~The interpreter then passes the acquired filename to the ByteCodeloader class, where each line of the source file is broken down into an array of tokens. The first token will be used to create a bytecode subclass based on the content of the token (e.g ArgsCode subclass, LoadCode subclass…).~~

~~The rest of the tokens will be passed as an array to each of these bytecode subclasses, which will store the argument(s) based on its specific requirement (object initialization).~~

~~The initialized bytecode objects will then be put into any array, this array of bytecode objects will be passed to the constructor of the Program class.~~

~~The program class will call its address resolving function to convert the “label address” in some bytecode subclasses into an index-address format, for further processing.~~

~~After converting the addresses, the bycodeLoader class (loadCode()) returns a created Program object, which contains an array of all the bytecode objects created based on the source file.~~

~~The interpreter received the returned Program object, and passes it to the Virtual Machine constructor to create a Virtual Machine object.~~

~~The interpreter will then run the Virtual Machine Object’s embedded executeProgram() function, this function will retrieve the bytecode objects stored in the array of the Program object.~~

~~The Virtual Machine will run each of these bytecode object (run its exec()), based on the address and information store inside them. These bytecode objects will interact with the RunTimeStack class ONLY through the VirtualMachine Class.~~

~~This process will continue until HALT bytecode object was processed. Then program ends.~~

# Tank Game Class Descriptions

# Self-Reflection on the project

~~This project is a good practice of the interaction between abstract class –and its subclasses as well as the Stack data structure. It also helped me have a better understanding of how an interpreter works and how to read bytecode.~~

# Project Conclusion

~~The output program of this project successfully produced identical outputs (including DUMP function) as the provided samples.~~