CSC 413 Term Project Documentation

Summer 2023

Nhan Nguyen

923100929

CSC413.01

<https://github.com/csc413-SFSU-Souza/csc413-tankgame-nhannguyensf>

Table of Contents

[**1** **Introduction** 3](#_Toc142255563)

[1.1 Project Overview (focus of term project) 3](#_Toc142255564)

[1.2 Introduction of the Tank game (general idea) 3](#_Toc142255565)

[**2** **Development Environment** 3](#_Toc142255566)

[2.1 Version of Java Used 3](#_Toc142255567)

[2.2 IDE Used 3](#_Toc142255568)

[2.3 Special libraries used or special resources 3](#_Toc142255569)

[**3** **How to Build/Import your Project** 3](#_Toc142255570)

[3.1 Import/build the project: 3](#_Toc142255571)

[3.2 Commands to build the JAR. 4](#_Toc142255572)

[3.3 Commands to run the JAR. 4](#_Toc142255573)

[**4** **How to run your game. Rules and controls of the game.** 4](#_Toc142255574)

[**5** **Assumption Made** 4](#_Toc142255575)

[These assumptions helped shape the design and implementation of the virtual machine and interpreter. They provided clarity on the expected behavior and allowed for a more focused development process. It is important to note that these assumptions were specific to the project requirements and may differ if the project were to be expanded or tailored for different use cases. 5](#_Toc142255576)

[**6** **Class Diagram** 5](#_Toc142255577)

[**7** **Class Descriptions** 5](#_Toc142255578)

[**8** **Self-reflection** 5](#_Toc142255579)

[**9** **Project Conclusion/Results** 6](#_Toc142255580)

# Introduction

## Project Overview (focus of term project)

The project aims to develop a

## Introduction of the Tank game (general idea)

More detailed information and technologies used will be provided below.

# **Development Environment**

## Version of Java Used

Java 20 (Oracle JDK 20)

## IDE Used

IntelliJ IDEA 2023.2 (Ultimate Edition)

## Special libraries used or special resources

and where you got them from.

# **How to Build/Import your Project**

## Import and build the project:

Follow these steps:

1. Set up the Development Environment:
2. Ensure you have Java Development Kit (JDK) installed on your computer. You can download the JDK from the official Oracle website.
3. Install an Integrated Development Environment (IDE) such as Eclipse, IntelliJ IDEA, or NetBeans. These IDEs provide a user-friendly environment for Java development.
4. Download the Project: Obtain the source code for the Interpreter project. This can be done by downloading the project files from a repository.
5. Open the Project in your IDE:
6. Launch your chosen IDE and import the project into it. This process may vary slightly depending on the IDE you are using.
7. Create a new Java project and configure it to use the existing source code files.
8. Build the Project: Ensure that the project builds successfully without any errors. If there are any compilation errors, review the code and resolve them.
9. Go to File -> Project Structure.
10. Under the Project tab, ensure you have the appropriate JDK selected (JDK 20). If no JDK is available, you can add one by clicking "New..." and selecting the path to your installed JDK.
11. Hover over the "Mark Directory as" option on the "resources" folder to reveal a sub-menu. In the sub-menu, click on "Resources Root". The directory will now be marked as a resources folder.
12. Click the Build -> Build Project option to compile the game.
13. Ensure that the project builds successfully without any errors. If there are any compilation errors, review the code and resolve them.

## Commands to build the JAR.

List what Commands that were ran when building the JAR. Or Steps taken to build jar.

## Commands to run the JAR.

**java -jar csc413-tankgame-nhannguyensf.jar**

# **How to run your game. Rules and controls of the game.**

After import/build the project, now we can run the application, follow these steps to run the project:

1. Go to Run -> Edit Configurations.
2. Click the + (plus) button and choose "Java Application" or "Application".
3. Name your configuration.
4. Set the "Main class" to Launcher class.
5. Click the green play button or Run -> Run "YourConfigurationName" to run the game.

# **Assumption Made**

During the design and implementation of the project, the following assumptions were made:

# They provided clarity on the expected behavior and allowed for a more focused development process. It is important to note that these assumptions were specific to the project requirements and may differ if the project were to be expanded or tailored for different use cases.

# **Class Diagram**

There is an UML diagram provides a visual representation of the project's class hierarchy, showcasing their relationships and interactions (like inheritance, aggregation, association, or dependencies) with each other.

Please see the picture of UML diagram in ***/documentation*** folder

# **Class Descriptions**

# **Self-reflection**

Throughout the development of the Interpreter project, I gained valuable insights into various aspects of software development and problem-solving. I had the opportunity to apply and enhance my knowledge of various programming concepts and technologies. Here are some key reflections on the project:

1. Programming Fundamentals: The project allowed me to reinforce my understanding of fundamental programming concepts such as variables, data types, control structures, and functions. Assignment 01 was my good start in Java language after a long time. And this Assignment 02 provided me an opportunity to learn more and practice those concepts again.
2. Modularity and Extensibility: The project's design emphasized modularity and extensibility, enabling the addition of new bytecode instructions and potential language expansions. The use of interfaces and class inheritance facilitated the integration of new features and improved the project's flexibility.
3. Object-Oriented Design: The project emphasized the use of object-oriented programming principles. It helped me comprehend the significance of encapsulation, inheritance, and polymorphism in building modular and extensible software systems. OOP helped in organizing the codebase and promoting code reusability.
4. Error Handling: The project implemented comprehensive error handling mechanisms to detect and handle exceptions during bytecode execution. This ensures informative error messages are provided, aiding in identifying and resolving issues within the program.
5. Version Control: I utilized version control systems like Git and use Github to manage the project's source code. It facilitated collaboration, allowed me to track changes, and provided a safety net in case of errors or regressions. Version control played a crucial role in maintaining code integrity and enabling easy code sharing.

In conclusion, the project served as a platform for my continuous learning and exploration of new concepts. It helped me review my knowledge about Java programming language to prepare for the upcoming term assignment. Additionally, it motivated me to stay updated with programming best practices and emerging technologies.

# **Project Conclusion/Results**

In conclusion, the project has successfully implemented essential components, such as bytecode instructions, the virtual machine, and the runtime stack. The project demonstrates effective code translation, accurate execution of bytecode instructions, and proper handling of runtime stack operations.

Assumptions were made during the project's design and implementation, focusing on the stack-based architecture of the bytecode language and the core functionality of the virtual machine and interpreter. While the project covers the essential aspects, further iterations may be necessary to expand language features and optimize performance.

Also, the implementation discussion highlights the design choices made, such as modular bytecode instructions, an interpreter for code translation, and a virtual machine for bytecode execution. A UML diagram illustrates the class structure and relationships within the project, providing a visual representation of its components. Project reflection encompasses the development process, challenges faced, lessons learned, and potential improvements. It offers an opportunity to evaluate the project's strengths and areas for enhancement, providing insights for future iterations.

In conclusion, the project has achieved its goal and provides a robust execution environment, accurate translation of high-level code into bytecode, and efficient execution of bytecode instructions. The project serves as a solid foundation for further language development and expansion. Overall, the Interpreter project served as a valuable learning experience in implementing algorithms, object-oriented design, and error handling. It demonstrated the successful application of these concepts to create a functional application.