International University, HCMC National University
School of Computer Science and Engineering

Final Report

BATTLESHIP

Submitted by.

No.	Name	Student ID	Contribution (%)
1	Phan Bảo Trân	ITDSIU21125	50
2	Lê Huỳnh Nhã Nguyên	ITDSIU21058	50

Course name: Algorithms and Data Structures

Professors: Dr.Vi Chi Thanh

TABLE OF CONTENTS

Abstract

This report delves into the creation of a 2D game adaptation of Battleship. The goal of this project is to develop a simple video game version of Battleship that can be played on computers. This version is inspired by the traditional Battleship game, which was originally played with pencil and paper or as a physical board game.

The development process utilizes the Java Programming Language and its extensive libraries, with Eclipse serving as the Integrated Development Environment (IDE). This setup allows us to effectively implement the algorithms and data structures concepts that we have learned throughout our course.

In this paper, we will analyze the essential principles and key mechanics of the Battleship game. We will build simple algorithms, focusing on fundamental structures such as arrays. Additionally, we will develop a viable software architecture for the game and explore various individual solutions and their alternatives.

Furthermore, the report will discuss the actual code developed for this project, which will be detailed in the following report. This subsequent report will also include UML diagrams that outline the design and structure of our code, as well as the implementation results, providing a comprehensive view of our development process and the functionality of the final product.

Throughout this project, we aim to demonstrate not only the practical application of theoretical knowledge but also our problem-solving and software engineering skills. By integrating these elements, we strive to create a functional and engaging digital version of Battleship.

Task Contribution

No.	Name	Responsibility
1	Phan Bảo Trân	- Implementing 6 classes : BattleShipAI, SimpleRandomAI, SmartAI, Ship, Game, GamePanel. - Writing the report. - Creating presentation slides. - Creating UML diagrams.
2	Lê Huỳnh Nhã Nguyên	- Implementing 5 classes : Rectangle, Marker, SelectionGrid, Position, StatusPanel. - Writing the report. - Creating presentation slides. - Creating UML diagrams.

CHAPTER 1: INTRODUCTION

1.1. Game Overview

Battleship (also referred to as **BattleShips** or **Sea Battle)** is a strategic guessing game designed for two players. The game is played on grids (either paper or board) where each player marks the positions of their fleet of ships, keeping these locations hidden from their opponent. Players take turns calling out coordinates in an attempt to "hit" and ultimately destroy the opponent's ships. The primary goal of the game is to obliterate the other player's entire fleet.

Battleship has a rich history, originating as a pencil-and-paper game during World War I. In the 1930s, it was commercially produced as a pad-and-pencil game by various companies. In 1967, Milton Bradley introduced it as a plastic board game. Over the years, Battleship has evolved, becoming available in electronic formats, video games, mobile apps, and even inspiring a feature film. It is also noteworthy as one of the earliest games adapted into a computer game.

1.2. Project Objectives

Our project seeks to create a simplified version of the classic game Battleship. This adaptation will employ data structures written in Java, and leverage JavaX extensions to enhance visualization and provide a more engaging game background. Through this project, we aim to demonstrate our proficiency in Java programming and our ability to design and implement algorithms using arrays.

Additionally, this project offers a valuable opportunity for us to apply and showcase our understanding of fundamental programming concepts and techniques. By building this game, we will explore various aspects of software development, including user interface design, game logic implementation, and the efficient use of data structures.

We will also focus on creating a user-friendly experience, ensuring that the game is both enjoyable and accessible. This involves not only the technical aspects of coding but also considering the overall user interaction and aesthetic appeal of the game. Through this comprehensive approach, we intend to deliver a functional and visually appealing version of Battleship that highlights our skills and knowledge gained throughout our coursework.

Moreover, the project will involve analyzing different approaches and alternative solutions to common problems encountered in game development. By doing so, we aim to build a robust and scalable architecture for our Battleship game. The end result will not only be a testament to our technical abilities but also a reflection of our creativity and problem-solving skills in software development.

1.3. Programming language & Development tools

Our project employs **Java** as the primary programming language. For a multitude of reasons, we have selected **IntelliJ** as the main **Integrated Development Environment (IDE)** for implementing Java in our project. IntelliJ is a versatile IDE designed for developing applications in Java, as well as other programming languages such as C/C++, Python, and more. It features a core 'Workspace' and a plugin system that allows us to extend the IDE's functionality through various plugins, adding extra features and capabilities. Eclipse is a powerful development tool compatible with all major operating systems, including Windows, Mac OS, and Linux.

For creating diagrams, we utilize **diagrams.net**, a web-based diagramming tool that integrates seamlessly with Google Drive and Dropbox. Diagrams.net provides a comprehensive selection of shapes, visual elements, and a drag-and-drop interface, making it straightforward to create professional-looking diagrams and charts. The tool supports automatic layout configurations for flow charts, mind maps, and tree diagrams, streamlining the diagram creation process.

In terms of graphics for the project, we use **JavaX libraries**, which encompass Java Swing and Java AWT. Java Swing is a lightweight GUI toolkit that offers a rich set of widgets, including trees, image buttons, tabbed panes, sliders, toolbars, color choosers, tables, and text areas. It enables the creation of sophisticated and interactive user interfaces. Java AWT (Abstract Window Toolkit) is a more heavyweight API designed for developing GUI or windows-based applications in Java. AWT components are platform-dependent, ensuring that they are rendered according to the native look and feel of the operating system. The Java AWT provides essential graphics and imaging tools, such as shape, color, and font classes, which are crucial for developing the visual aspects of our game.

The combination of Java's powerful programming capabilities, IntelliJ's extensive development features, and the sophisticated graphical tools provided by JavaX libraries enables us to create a visually appealing and functional version of the Battleship game. By leveraging these technologies, we aim to deliver a high - quality software project that demonstrates our proficiency in programming, software design, and user interface development.

CHAPTER 2: GAME PLAY

2.1. Game Rules

At the outset of the game, each player discreetly places their ships on their respective grid. Ships can be positioned either horizontally or vertically but must not overlap. Players can change the direction of the next ship to be placed by pressing the Z key on the keyboard.

The game screen features two grids: one for the player and one for the opponent (computer). On the player's grid, ships are arranged and the shots made by the opponent are recorded. Conversely, the second grid is used by the player to record their own shots at the opponent's ships. When a player makes a successful hit on an opponent's ship, the shot is marked red. If the shot misses, it is marked blue.

The objective of the game, as indicated by its nature, is for each player to guess the locations of the opponent's ships and destroy all of them to win the game. Players make guesses by selecting a cell to

attack at random. The subsequent move is influenced by the outcome of the previous shot—whether it was a hit or a miss. The first player to sink all of the opponent's ships emerges as the winner. After the game concludes, players have the option to rearrange their ships and start a new game.

2.2. Character

This game features two players: you and the computer, which plays automatically using AI. We have designed four types of ships, each with varying lengths. These include 1 Battleship (length = 5), 1 Submarine (length = 4), 2 Cruisers (length = 3), and 1 Small Boat (length = 2).

2.3. Feature descriptions

In this game, the player can play with the computer built with a system of algorithms that can make it choose the next move smartly.

CHAPTER 3: DATA STRUCTURES

This project utilizes ArrayList as the primary data structure, chosen for its versatility and ease of use in managing collections of objects. Every game element, including ships, fields, markers, positions, and grids, is implemented using ArrayLists. This decision allows the algorithm to efficiently iterate through each object sequentially, providing a clear and manageable approach to handling the game's components.

The ArrayList data structure is particularly advantageous because it dynamically resizes, which is useful for adding or removing elements without the need for manual resizing. This flexibility ensures that the game can handle varying numbers of ships and shots efficiently. During the execution of the algorithm, the ArrayLists are populated and cleared in each loop iteration, ensuring that the game's state is updated accurately with each move.

To maintain simplicity and clarity, the project deliberately avoids using more complex data structures like graphs. Incorporating graphs would significantly complicate the algorithms, adding unnecessary complexity to the project. By keeping the data structures straightforward, we ensure that the focus remains on core gameplay mechanics and logic.

Moreover, the decision to use ArrayList supports the goal of making the code easy to understand and maintain. The straightforward approach means that there is no need for complex design patterns, which can sometimes obfuscate the underlying logic. This makes it easier for other developers to read and contribute to the codebase, facilitating collaboration and future enhancements.

Despite the simplicity, the use of ArrayLists does not compromise the functionality of the game. The data structures are robust enough to handle all necessary operations, such as placing ships, recording shots, and updating the game state. This balance of simplicity and functionality is a key aspect of the project, ensuring that it is both effective and easy to manage.

Additionally, future iterations of the project could explore optimizing performance by tweaking how ArrayLists are used, potentially incorporating advanced features of Java's Collection Framework as

needed. However, for the current scope, the focus on ArrayList ensures that the project meets its objectives without unnecessary complexity.

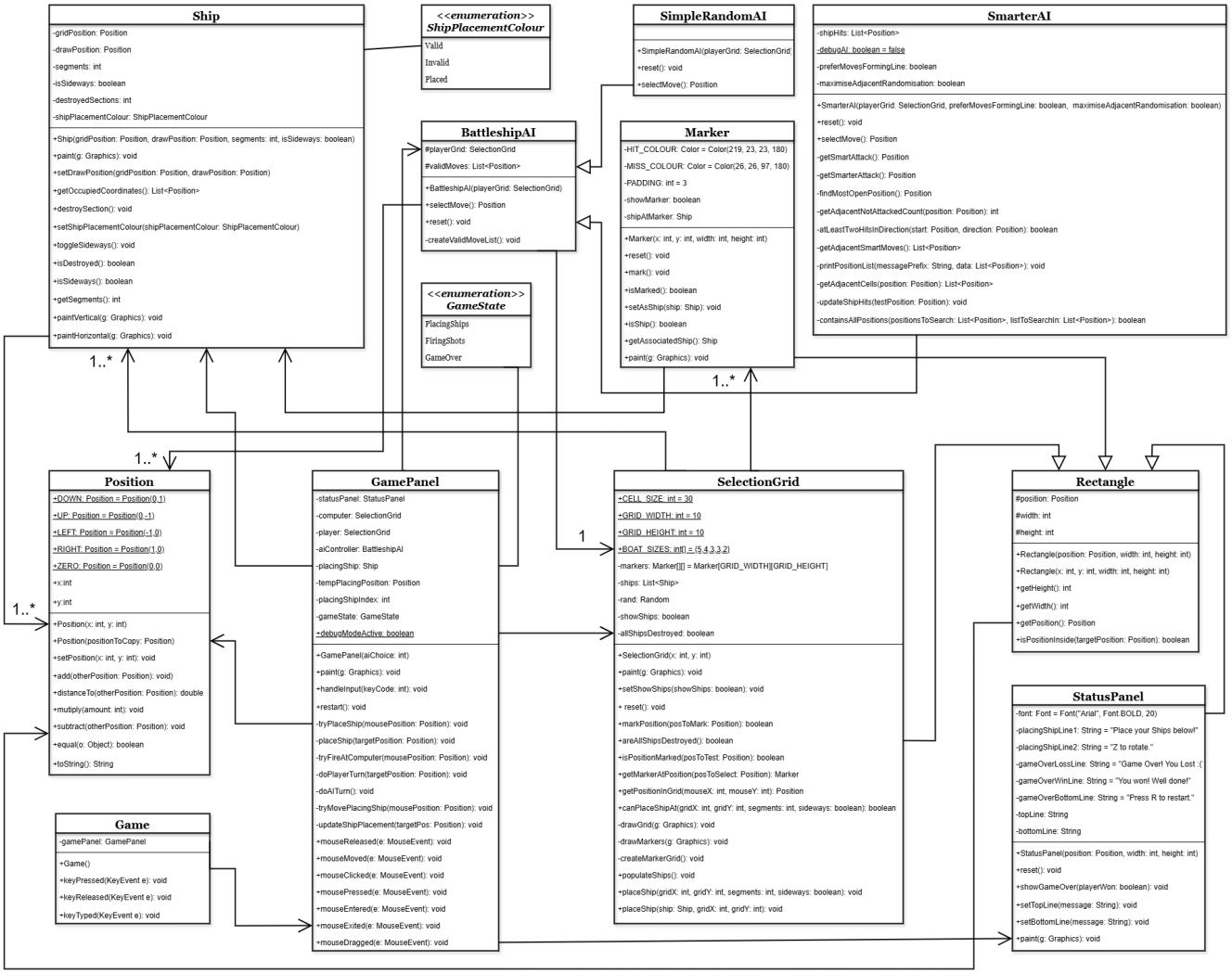
In conclusion, by utilizing ArrayList as the main data structure, the project achieves a balance between simplicity and functionality. The clear and manageable codebase, free from complex design patterns, allows for straightforward development and easy maintenance, ensuring a robust implementation of the Battleship game.

CHAPTER 4: GAME DESIGN AND ARCHITECTURE

4.1. List of Classes

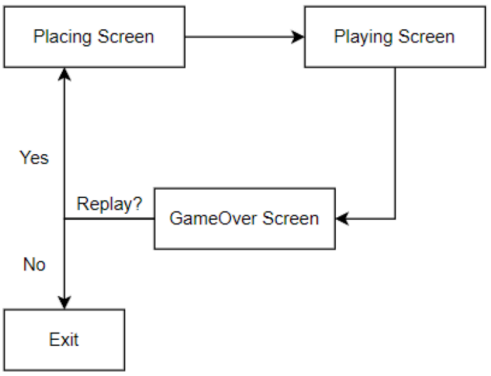
No	Class Name	Description
1	Position	Manages the coordinates used for predicting positions.
2	Ship	Create a ship with its position, size and status.
3	Rectangle	Defines simple rectangles used to create grids, including their position, width, and height.
4	Marker	Represents colored rectangles indicating whether a position is hit or missed.
5	SelectionGrid	Stores ships and uses a grid of markers to show hit/miss detection.
6	StatusPanel	Displays messages to the player.
7	BattleshipAI	Template class to be extended to provide actual AI behavior.
8	SimpleRandomAI	A very simplistic AI that does not use any commonsense but selects the first random moves from a shuffled list.
9	SmarterAI	An AI that searches randomly until it finds ships.
10	Game	The main class to run the game.
11	GamePanel	Manages the interaction and state information between game elements.

4.2. Class Diagrams



You can see the general diagram more clearly **here**.

4.3. Game Structure



4.4. Graphic

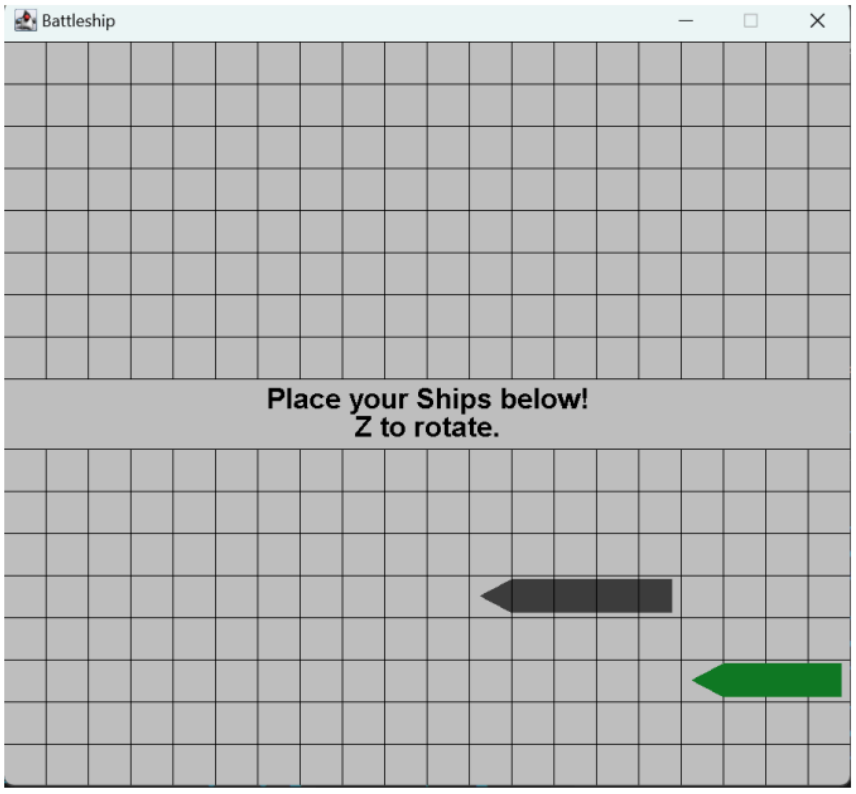
This project utilizes the **JavaX** library, which is a built-in Java library, to assist users in creating simple graphics. It includes functionalities for drawing, creating panels, and displaying messages.

CHAPTER 5: IMPLEMENTATION RESULTS

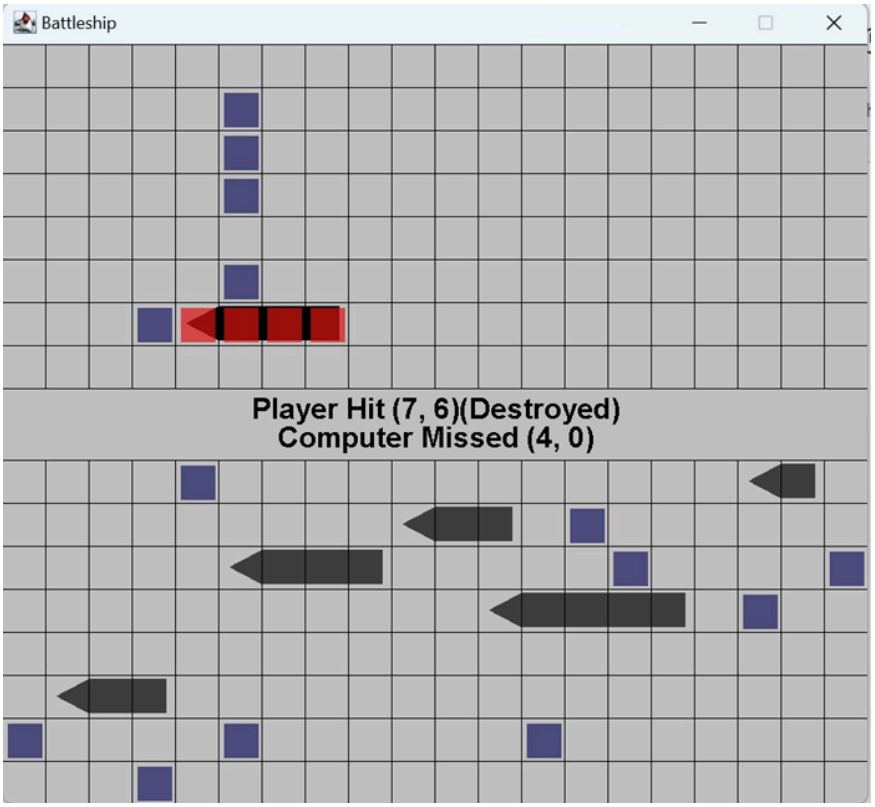
5.1. Control instruction

- At *Placing* Screen:
  - Move the mouse to select ship positions.
  - Click on the grid to place the ship
  - Press Z to rotate the ship
- A*Playing* Screen:
  - Click on the grid to fires shots
- At *GameOver* Screen:
  - Press ESC to exit
  - Press R to replay

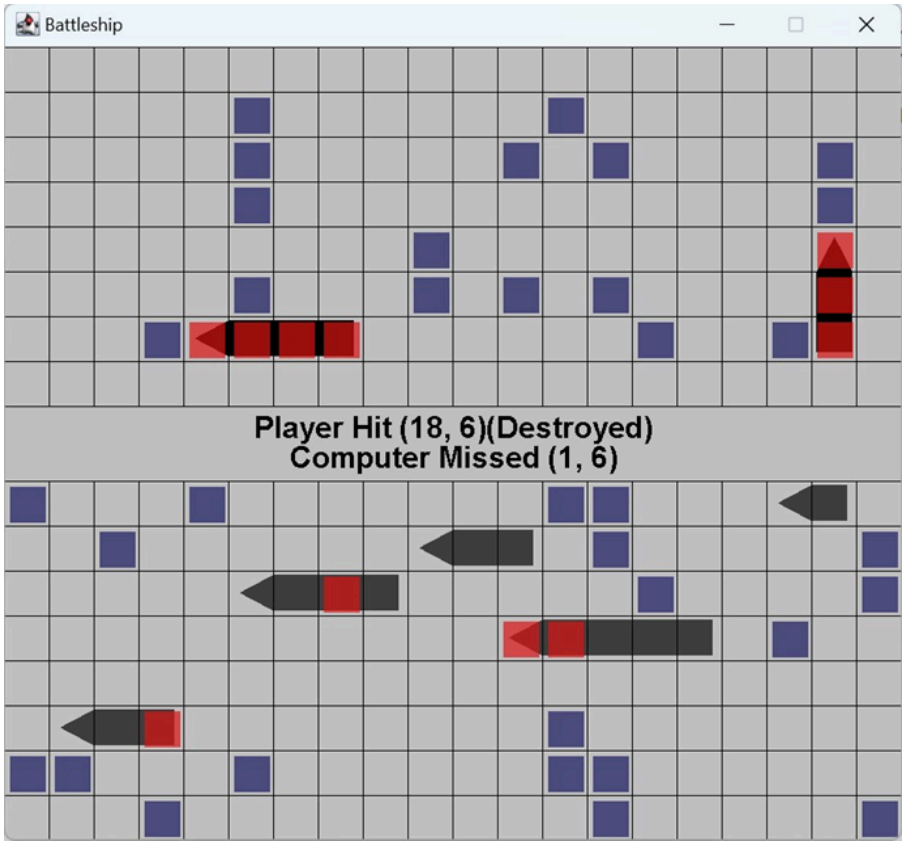
5.2. Screenshot



Playing Screen



Playing Screen



Playing Screen

CHAPTER 6: CONCLUSION

The project successfully implemented ship positioning, battle mechanics, an AI opponent, and basic graphics. It utilized **data structures** like ArrayList and List from the course. However, the graphics are simple, and the code could be more organized.

6.1. Limitations

Though this is a complete game, the graphics seem to be so simple and the code is quite confusing.

6.2. Future Plans

- Improve graphics with advanced libraries/tools like JavaFX, Unity, or Unreal Engine.
- A booster system to increase the attraction and diversity of the game such as: *double-shot* for shooting twice, *railgun* for destroying a whole row,..
- A PvP system for players to play together through a LAN network or Internet.
- Implement a score-based ranking system to enhance competitiveness.

A. Git: Link To The Game Project

https://github.com/tranphan2910/DSA_BattleShip

B. Reference List

(n.d.). Diagram Software and Flowchart Maker. https://www.diagrams.net/ Hutchins, E.

(n.d.). *Battleship (game)*. Wikipedia.

https://en.wikipedia.org/wiki/Battleship_(game)