Computer Network and Design Laboratory
PROJECT REPORT

Meryem Kılıç – 2121251018

23/05/2025

Battleship Game - Project Documentation Report

1. Project Overview

Battleship Game is a network-enabled version of the classic board game, developed using the Java programming language. The project is built on a client-server architecture, providing a platform where two players can play against each other.

2. Technical Infrastructure

- **Programming Language:** Java 8+
- **Architecture:** Client-Server
- **Network Protocol:** TCP/IP
- **GUI Framework:** Java Swing
- **Build Tool:** Maven
- **Design Pattern:** MVC (Model-View-Controller)

3. Project Structure

The project consists of the following main packages:

3.1 Common Package (com.mycompany.savasgemisi.common)

- **Message Classes**
- Message.java: Handles the basic message structure and serialization for network communication
- GameMessage.java: Manages game-specific messages like moves, hits, and game state updates
- ConnectionMessage.java: Handles connection-related messages such as join requests and disconnections
- **Model Classes**
- Board.java: Represents the game board, manages ship placements and hit/miss tracking
- Ship.java: Defines ship properties and state (position, size, damage status)
- Position.java: Handles coordinate system and position validation
- GameState.java: Maintains the current state of the game (player turns, game phase)
- **Utility Classes**
- Constants.java: Stores game configuration and constant values
- GameUtils.java: Provides helper functions for game operations

3.2 Server Package (com.mycompany.savasgemisi.server)

- **Main Classes**
- GameServer.java: Core server implementation, manages server lifecycle and client connections

- ServerGUI.java: Provides server administration interface and connection monitoring
- **Management Classes**
- ClientHandler.java: Manages individual client connections and message routing
- GameSession.java: Coordinates game sessions between two players
- PlayerManager.java: Handles player registration and session assignment
- **Game Logic**
- GameLogic.java: Implements core game rules and turn management
- ShipPlacement.java: Handles ship placement validation and randomization
- GameValidator.java: Validates game moves and state changes

3.3 Client Package (com.mycompany.savasgemisi.client)

- **Main Classes**
- GameClient.java: Manages client-side game logic and server communication
- ClientGUI.java: Provides the main game interface for players
- **Interface Components**
- GameBoard.java: Renders the game board and handles player interactions
- ShipPanel.java: Displays ships and their states
- ControlPanel.java: Provides game controls and status information
- **Communication Classes**
- ServerConnection.java: Manages network connection to the server
- MessageHandler.java: Processes incoming messages from the server
- GameController.java: Coordinates between GUI and game logic

4. Network Architecture and Threading

4.1 Server-Client Communication

- **Server Side**
- Listens on specified port for client connections
- Maintains separate connections for each client
- Broadcasts game state updates to connected clients
- Handles client disconnections gracefully
- **Client Side**
- Establishes connection to server
- Sends player moves and actions
- Receives game state updates

- Maintains connection state

4.2 Threading Implementation

- **Server Threads**
- Main server thread for accepting connections
- Dedicated thread for each client connection
- Game session thread for managing active games
- GUI update thread for server interface
- **Client Threads**
- Main GUI thread for user interface
- Network thread for server communication
- Game logic thread for processing moves
- Event dispatch thread for UI updates

4.3 Thread Synchronization

- **Server Synchronization**
- Thread-safe message queue for client communications
- Synchronized game state updates
- Atomic operations for game moves
- Lock mechanism for shared resources
- **Client Synchronization**
- Thread-safe message handling
- Synchronized GUI updates
- Atomic game state changes
- Event queue for user interactions

5. Features

- Network multiplayer game support
- Graphical user interface
- Random ship placement
- Intuitive game board
- Real-time game state updates
- Restart game option

6. Game Rules

- 10x10 game board

- Ships:
- 1 ship of 5 units
- 1 ship of 4 units
- 2 ships of 3 units
- 1 ship of 2 units
- Turn-based shooting
- Visual indication of hits and misses
- First player to sink all opponent's ships wins

7. Installation and Execution

1. Java 8 or higher version required
2. Build with Maven:

text

Apply to SClient.java

```
mvn clean package
```

3. Run the JAR file:

text

Apply to SClient.java

```
java -jar target/savasgemisi-1.0-SNAPSHOT.jar
```

8. Usage Scenarios

1. **Starting the Server**

- Select option "1" from main menu
- Enter port number
- Wait for client connections

2. **Client Connection**

- Select option "2" from main menu
- Enter server IP and port information
- Connect using the "Connect" button

9. Security Measures

- Port number validation
- Connection state monitoring
- Game state synchronization

This documentation provides a comprehensive overview of the project's architecture, focusing on the main responsibilities of each class and the network/threading implementation details.