

Developer Manual (Windows + IntelliJ IDEA)

Environment Overview

- **Operating System:** Windows 11.
- **JDK:** Oracle OpenJDK 21.0.4 (LTS).
- **IDE:** IntelliJ IDEA Community 2024.2 or newer.
- **Testing:** JUnit 5.8.1, you need to add the JUnit module in this IntelliJ project (Steps mentioned in the section Testing below).

Project Structure

```
1 | Implementation/
2 |   └ src/           # Game source code (entry: Main.java)
3 |     └ model/, pages/, ...
4 |   └ Test/          # JUnit 5 tests mirroring main packages
5 |   └ out/           # IntelliJ default build output
6 |   └ archive/, replay/ # Saved games and replays
```

Opening the Project in IntelliJ IDEA

1. Launch IntelliJ IDEA → `File > open...` → select the repository root (`Implementation`).
2. When prompted, choose **Open as Project**. IntelliJ auto-detects `src` as Sources and `Test` as Tests. You can verify via `File > Project Structure`.
3. Configure SDK: `File > Project Structure > Project SDK` → select installed **JDK 21**.

Building via IntelliJ

1. Use `Build > Build Project` to compile both main and test sources.
2. IntelliJ outputs compiled classes to `out/production/Code` (main) and `out/test/Code` (tests) by default.

Running/Debugging the Game

1. Create a run configuration: `Run > Edit Configurations... → Application`.
 - `Main class: Main`.
 - Program arguments: *(leave empty)*.
2. Click **Run** to launch; the console shows the text-based menu (`startNewGame`, `loadGame`, `watchReplay`, `exit`).
3. For debugging, set breakpoints anywhere in `src/model` or `src/pages` and click **Debug Main**. Standard debugger tools (step into/out, evaluate expression, watches) are available.

You may also directly enter class `Main` under `/src` and click the button of the `main` method to run.

Running Unit Tests (IntelliJ)

Import JUnit library

1. Go to `File > Project Structure > Project Settings > Libraries`, click `+` > `From Maven`.
2. Enter `org.junit.jupiter:junit-jupiter:5.8.1` and press `OK`.

Run Unit Test

1. Right-click the `Test` directory or any specific test class (e.g., `Test/model/BoardTest.java`) and choose **Run 'Tests in ...'**.
2. IntelliJ's test runner reports pass/fail counts and stack traces. Use **Rerun Failed Tests** for quick iterations.

Coverage in IntelliJ

1. Choose **Run 'Tests in ...' with Coverage** to collect coverage data.
2. The IDE displays line and class coverage overlays inside editors.

Debugging Tips

- Utilize conditional breakpoints when diagnosing complex move validations in `model/MovingValidator.java`.
- The textual UI reads from `scanner`. To simulate scripted inputs during debugging, prepare a `.txt` file with commands and redirect standard input in the run configuration (`Modify options > Redirect input`).

Launch Checklist

1. Ensure JDK 21 SDK configured.
2. Import project and mark source/test folders.
3. Create run configuration and add JUnit to the project.
4. Verify tests pass (either via IDE) before committing.