Developer Manual of Monopoly

Written by YEUNG Tsz Lok 22076383d

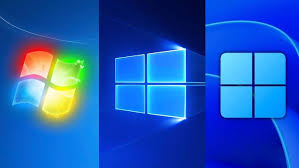
Brief Introduction:

This is Daniel from Group43. In this Manual, we will go through the expected specifications for the software to be run at, i.e. depiction of the required Operating System (OS) platform, programming language, the language package version (Dev Kit) , intended Integrated Development Environment (IDE) used for the Monopoly game.

By any means, here’s a brief Table of Contents.

1. OS -----------------------------------------------pg.2
2. Programming Language----------------------pg.3
3. Development Kit-------------------------------pg.4
4. IDE------------------------------------------------pg.4
5. Simple setup (Lab computer example)----pg.5-8
6. **Operating System**

Our software is designed to run on the Windows operating system, specifically versions 10 and above. Our choice allows for broad compatibility with a range of hardware configurations, but most importantly commonly used in development and gaming environments, and general purposes. Windows just simply has a user-friendly interface and stable support for Java applications, making sure the Monopoly game can be run at ease.



1. **Programming Language**

The Monopoly game is developed using the Java programming language. The language's strong adherence to object-oriented programming (OOP) principles promotes modular design, we are enabled to create reusable code components that can be easily maintained and extended.

Java’s rich set of libraries and frameworks further enhances development efficiency, which support some pre-built functionalities that helping our coding process faster. Java as well displays strong type-checking and exception handling capabilities contribute to building a more reliable and error-free application. Overall, Java's performance, combined with its OOP features, makes us decide to choose it.

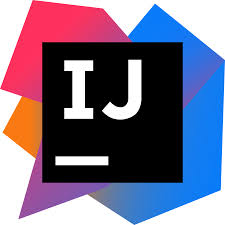


1. **Java package/Development Kit version**

We are running JDK-22. It has high compatibility and language syntax can be written cleaner and more concise. We also introduced some advanced functionalities incorporated in our software.

1. **Integrated Development Environment**

IntelliJ IDEA is the IDE we selected. We chose IntelliJ due to its powerful code assistance features, fair code layout and analysis, with refactoring tools.



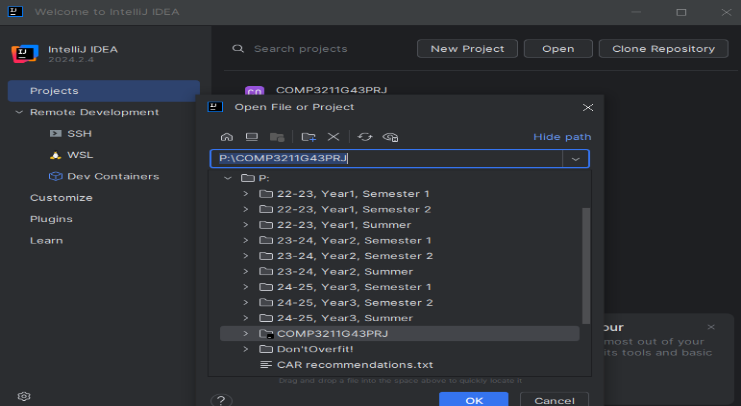
1. Setup for running the software / in debugging mode

We tested >=3 times setup with PQ lab computers in 6th floor. The following procedures are mandatory for setup.

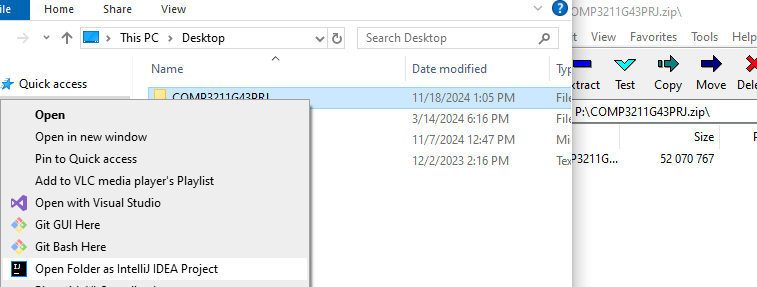
1. Open IntelliJ

In order to run the application, (Assuming you already have IntelliJ installed on your computer) we need to first execute IntelliJ IDEA from JetBrains, our ideal IDE for the developed game software.

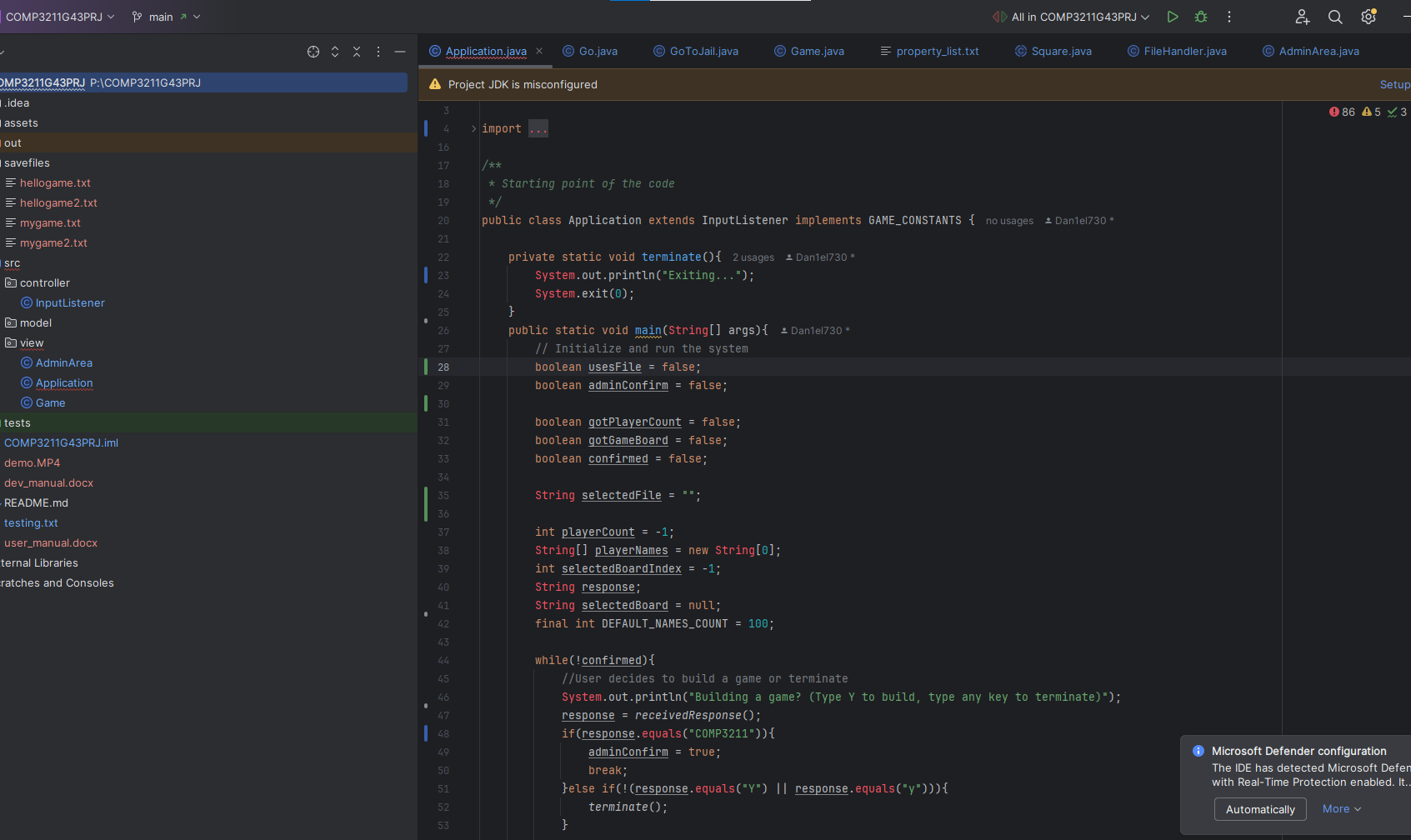




Search IJ IDEA by Windows key, then select our project folder to open as Project

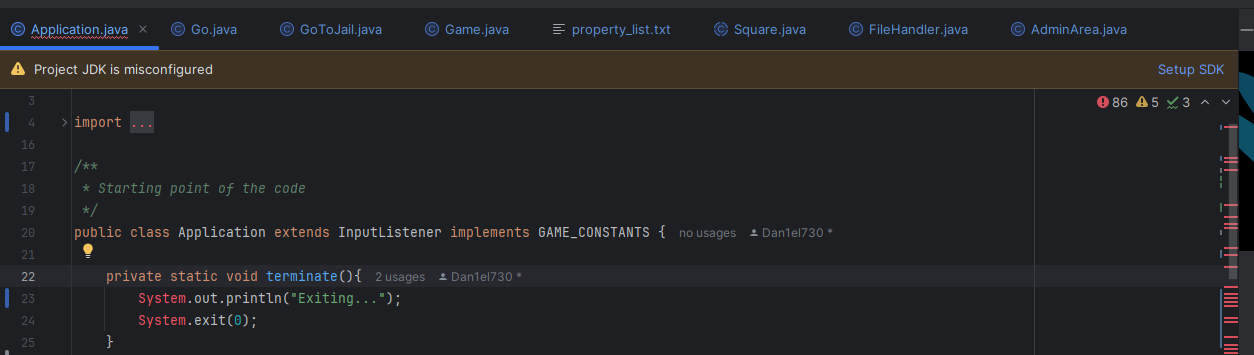


Or directly unzip the folder, then Open Folder as IJ IDEA

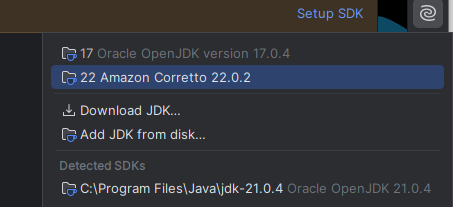


The Project Editor should be displayed afterwards as shown

1. Configuring JDK

Next, with the IJ opened up, you may automatically use the IJ function to configure JDK to version 22

JDK can be misconfigured, Setup JDK to fix that



Choose version 22

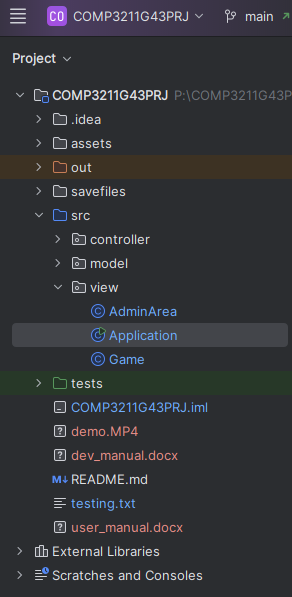


If the JDK package is corrupted/unfound,redownload it with IJ’s help, then you may reopen the Project with IJ again to finish the setup.  
The “Project JDK is misconfigured” should disappear!

Upon downloading/configuring the JDK, you may get ready to run the application.

1. Running the ‘main program’ (Application.java)

Our architecture supports that our program start from an Initialization Loop, and it starts from “src/view/Application.java”. Where the game configuration is supposed to be started, here is the supposed access path to the class by Editor.



**1**

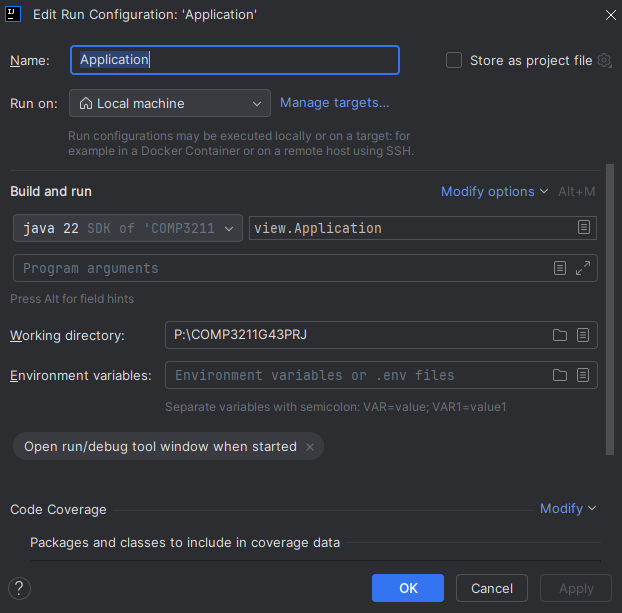
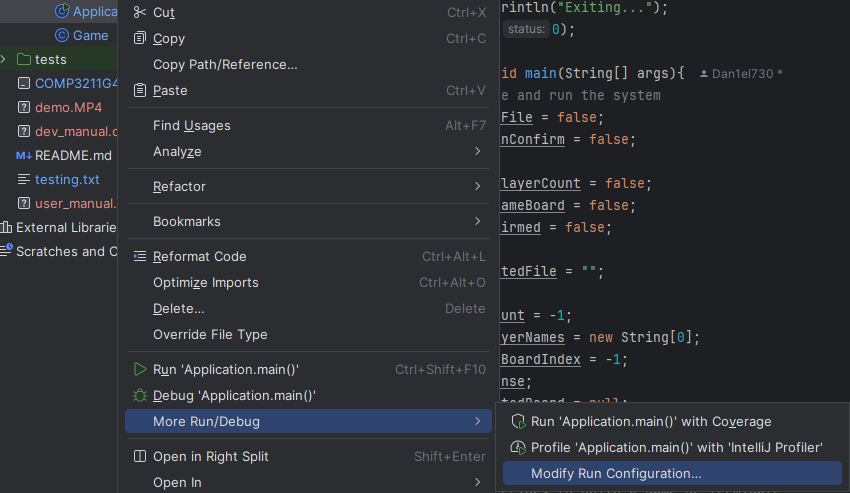
**2**

**3**

**4**

shown path from source directory to access Application.java

In general, if there no Run Configuration in your Editor side, it is a good practice to help your environment add in one for executing the main code of the whole software.



1. Running, Debugging, and Running Tests with Coverage.
2.  Running the game
3. User side

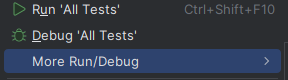
Simply use the Run button in IJ. To construct a game you are advised to use pre-written and produced-when-testing save files to quickly start a game. (Refer to user\_manual : Part 2b, page4)

1. Admin/Gameboard designer side

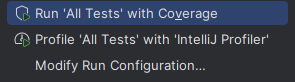
Use Run button again. Input the dev code : COMP3211 to modify stats about properties on boards and create new boards and save them into assets.

1. Debugging the game

You can use the Debug button in IJ, add in breakpoints by creating red dots in proper code lines if desired.

1. Running tests in coverage

Look for a green

 directory inside our main root

folder, right click then select

“More Run/Debug”,

“Run all in Coverage”. View coverage via model folder.