**Application Launch:**

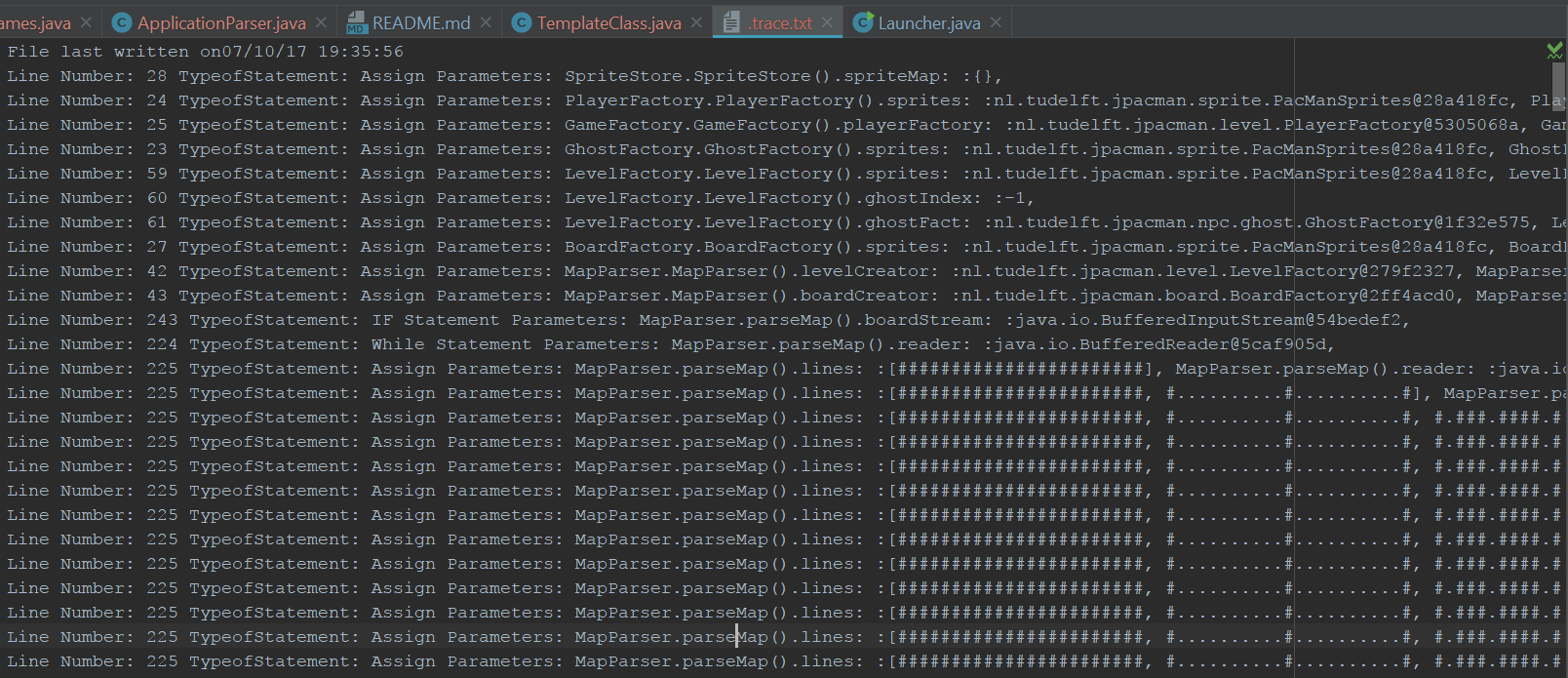
The main program to launch application is in **src/main/java/nl.tudelft.jpacman/Launcher.java**

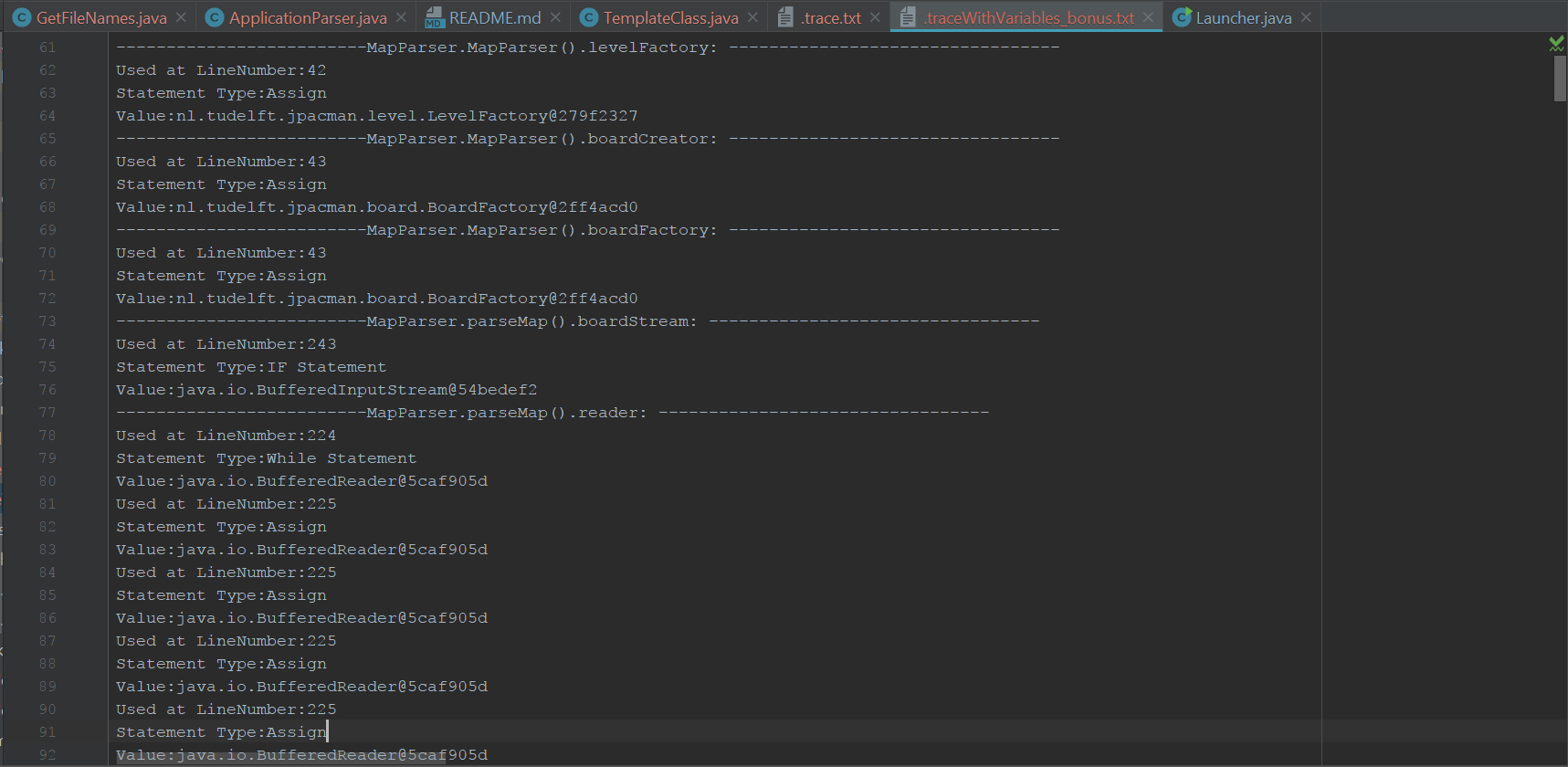
**Gradle and SBT builds:** Both Gradle and SBT builds have been included in the project

**Application Run:**

**Gradle** - Type Gradle run or Gradlew run command in the project folder will automatically run the application and start the Pacman game

**SBT** - SBT followed by run also will automatically run the application and start the Pacman game and the same **trace.txt** and **tracewithVariables\_bonus.txt** will be created

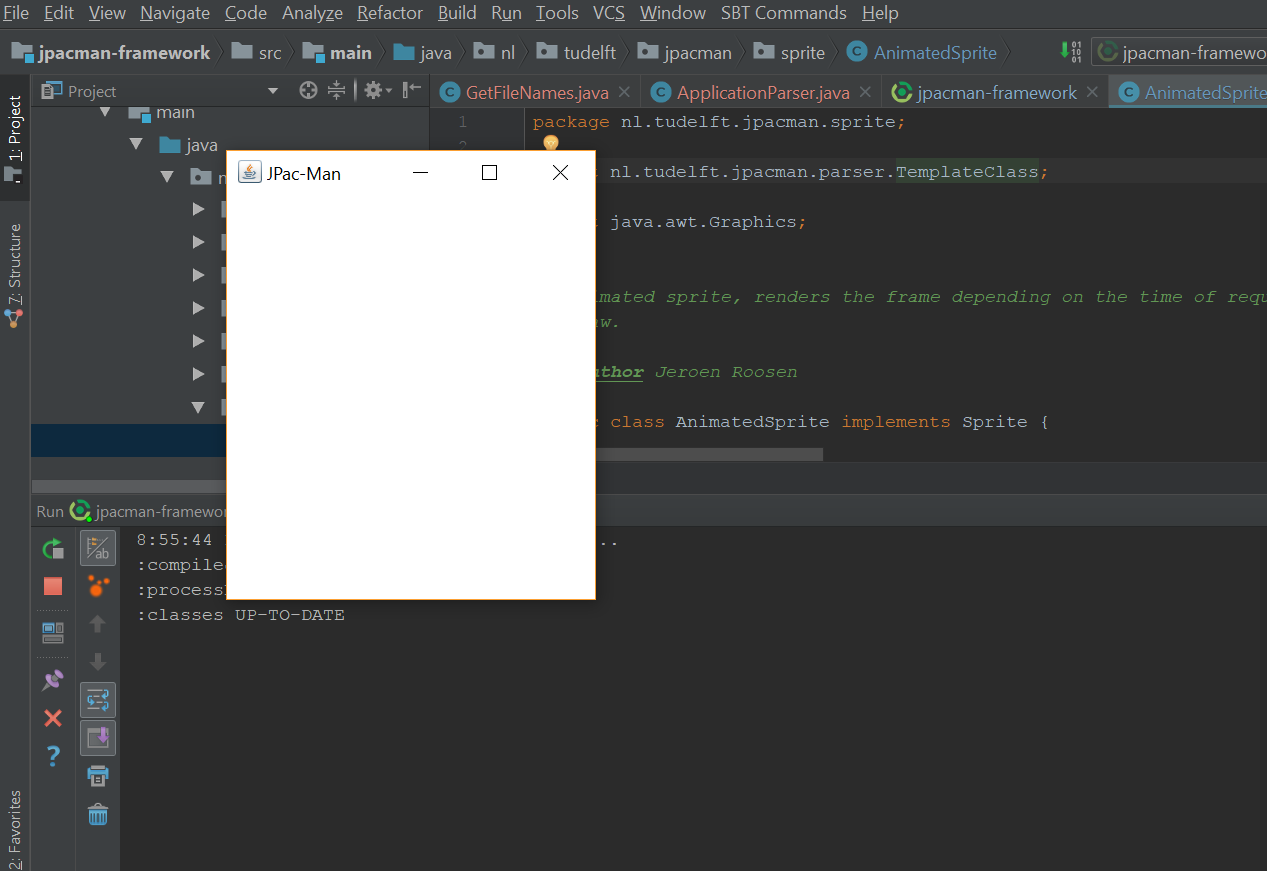
* **trace.txt file –** The trace.txt file gets generated in the project root folder which has the log of all the statements run in the program and the value of the variables created in the program
* 
* **tracewithVariables\_bonus.txt –** This file is a part of the additional 6% bonus points. This has the list of all the variables used in the code with its scope(class and method in which it is declared) and the lines in the code where it is used and what was the value of the variable during execution



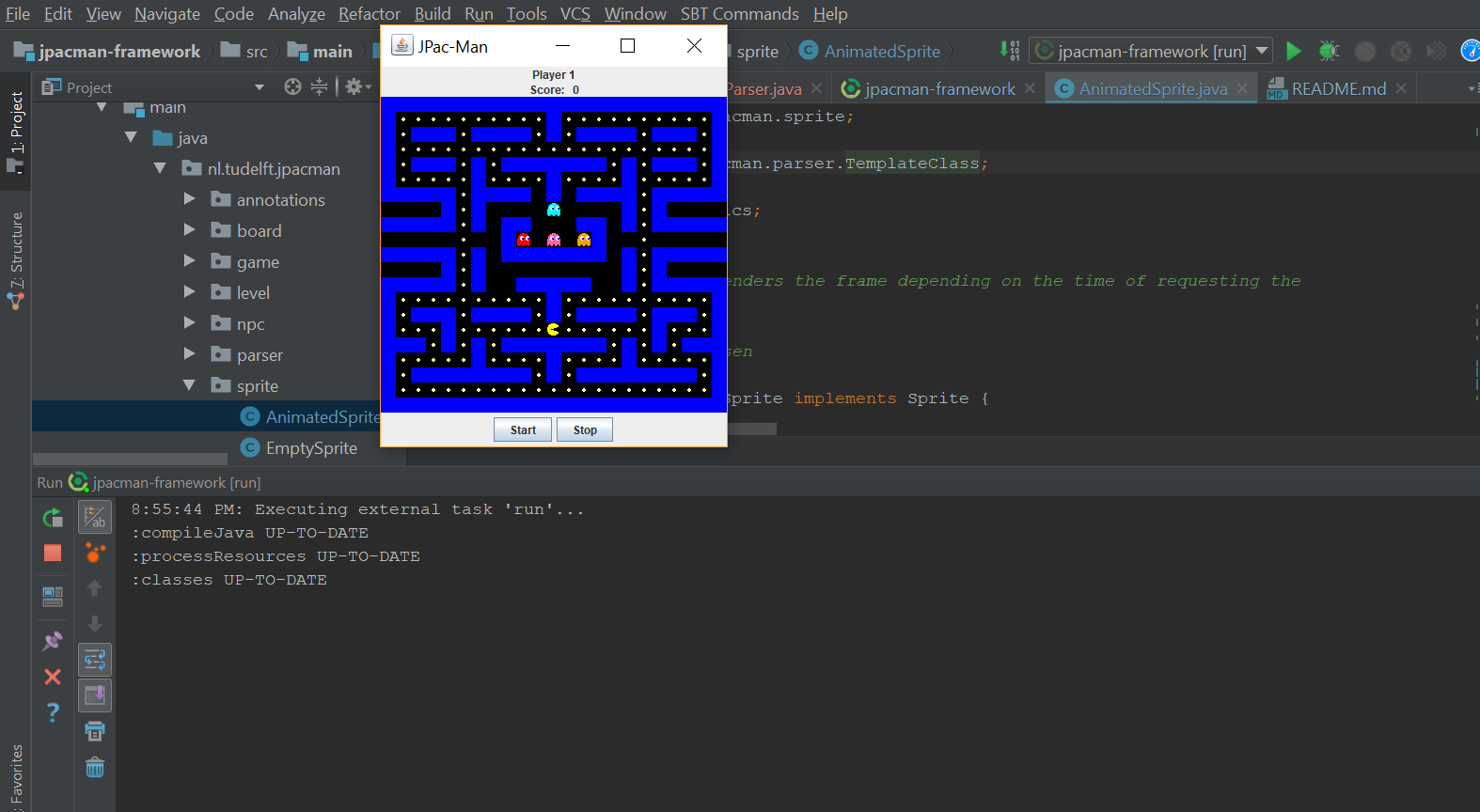
**Note to consider:**

Since the log statements(template.instrum()) code has been added after every line of code – the application has become very slow. And mine in a pacman game which gets executed in the fly and lot of lines of code keep running in the background when the game is being played with press of each key.

* Application is slow and for the game window to open it takes few more seconds than before(10-30 seconds)
* And the game window just opens as blank white image



* After a few more seconds the actual game appears



* But with all the logging happening in the background, the game is very slow. But there is no exception or error with the code which is running.

**Detailed Explanation of the implementation**

**---- First part for initial 5 % of the project**

**Parser Codes:**

Codes to parse the original java files and create java files with instrum logging statements are placed in the folder: **src/main/java/nl.tudelft.jpacman/parser**

Main code to parse – **ApplicationParser.java –** This code creates the new .java file with the instrum statements

**GetFileNames.java:**

This code gets the names of all the java files in the application and calls the Application Parser for each java code and new java codes are created with the instrumentation statements

**Old Java Files** – Are placed in the folder old\_source (Have changed them as .txt files so that no clash occurs in the code execution)

**When the application runs the new .java files with the instrumentation statement runs logging every statement which is then output finally in the trace.txt file**

**--- Second part for the bonus 5% of the project**

For the second part of the assignment instead of just printing the instrum statements into a text file.

Have created a List<ArgumentRecordings> which has a identifier field to distinctly get the scope and name of each variable in the code and has a List<ArgumentValues> which has the list of different places in the code the variable gets used or updated

In the output - **tracewithVariables\_bonus.txt**

**Each variable gets added in the file and every time the variable gets used or updated in some line in the code it is updated in the List<ArgumentRecordings> which is printing in the final file.**

**Junit Tests:**

Have Created 4 Junit tests in the src/test/java folder

**Junit Tests:** Similarly Gradle clean test and test command in sbt will automatically run the junit tests

**Java Monitoring Tools:**

Have used jconsole and jprofiler monitoring tools for this application.

**Jconsole: -** Launched jconsole from cmd and ran the application in intellij. In the jconsole application select the local application running to monitor the application using jconsole

**JProfiler: -** Launched Jprofiler seperately with the application running in intellij. In the Jprofiler select the running applciation to monitor

**Have attached the snapshots of these in the following folder**

**doc/Monitoring Tools - Screenshots**