README

This is a simple address book application. This application is forked from https://github.com/se-edu/addressbook-level4.git to my HW repo.

It performs add, delete, update and select operations on contacts in address book through a user interface.

I have created an associated GRADLE and SBT builds along with parsing logger information for this application.

This document has the following contents

1) GRADLE Build
   1.1) Build, run application and Generate Parsing Log information using Gradle build from INTELLIJ
2) SBT Build
   2.1) Build, run application and Generate Parsing Log using SBT Build from INTELLIJ
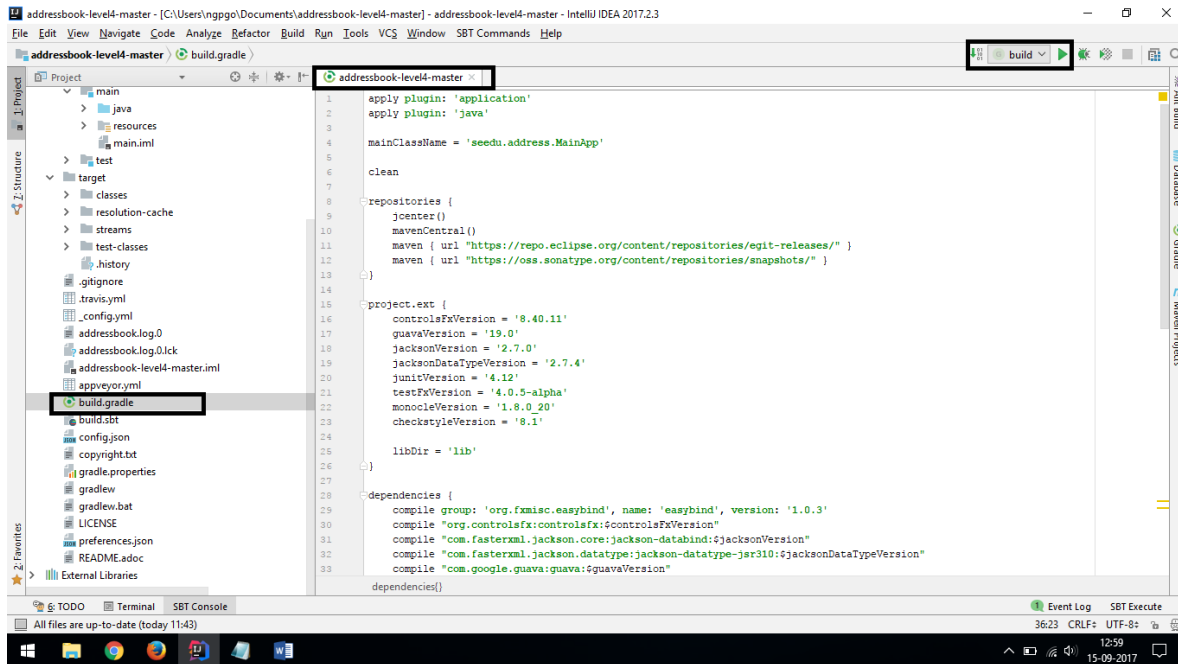3) AST Parsing program

Pre-requisites:

JDK version 1.8

Import Gradle changes, once the project has been opened/imported.

1) GRADLE

1.1) Build, run application and Generate Parsing Log information using Gradle build from INTELLIJ.

Once the project has been downloaded and imported as a project to INTELLIJ, kindly follow the below steps to BUILD and RUN the GRADLE file.
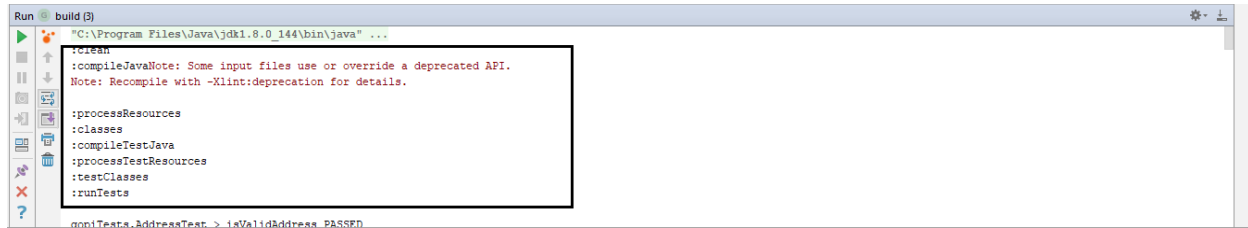

Choose the file build.gradle, where in a new window pops up. Once the window is opened click on the run radio button on the top right corner.

On the push of the run radio button a 'run window' pops up below the screen and the following tasks will be carried out (Screen shots attached below).

If the above option is not found, right click on the build.gradle file and choose run build option

1) Compile
2) JUNIT tests and the results will be displayed as passed or failed
3) The main app (UI) will be invoked (The app icon will be blinking on the taskbar) (Close the application to complete the parser logging)
4) The parser logging information will be printed on the run console as well as a Logging.txt file will be generated in the main directory.
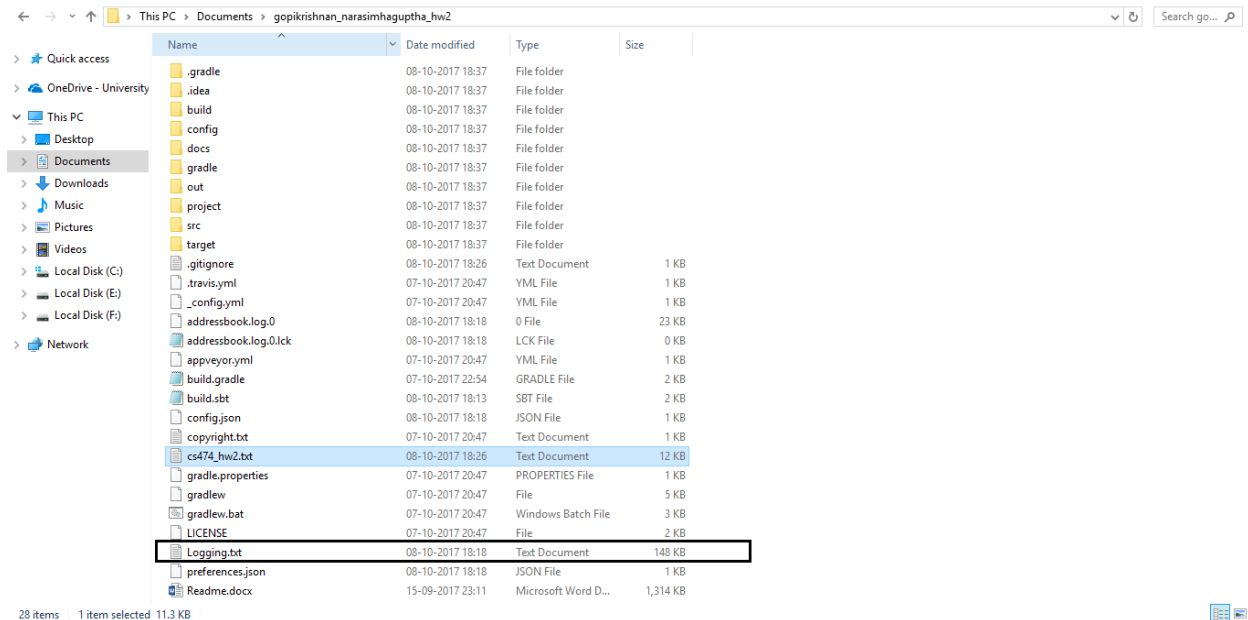
The Gradle build is functioned to do all the above-mentioned tasks when invoked from INTELLIJ.
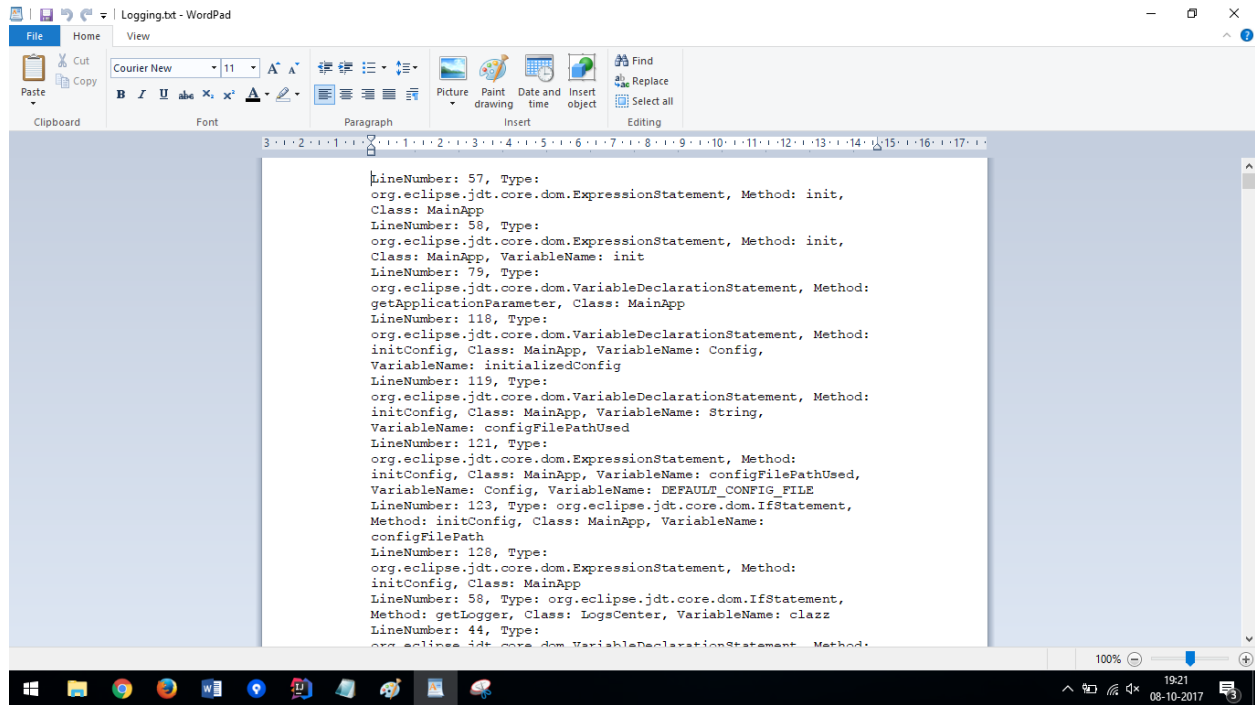
```
Run  build (3)
      "C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
      :clean
      :compileJavaNote: Some input files use or override a deprecated API.
      Note: Recompile with -Xlint:deprecation for details.

      :processResources
      :classes
      :compileTestJava
      :processTestResources
      :testClasses
      :runTests

      gopiTests.AddressTest > isValidAddress PASSED
```

The main app will be invoked, to get the complete parser logging, close the main app

```
LineNumber: 48, Type: org.eclipse.jdt.core.dom.ExpressionStatement, Method: getLogger, Class: LogsCenter
LineNumber: 91, Type: org.eclipse.jdt.core.dom.IfStatement, Method: addFileHandler, Class: LogsCenter, VariableName: fileHandler
LineNumber: 94, Type: org.eclipse.jdt.core.dom.ExpressionStatement, Method: addFileHandler, Class: LogsCenter
LineNumber: 49, Type: org.eclipse.jdt.core.dom.ExpressionStatement, Method: getLogger, Class: LogsCenter
LineNumber: 59, Type: org.eclipse.jdt.core.dom.ExpressionStatement, Method: readJsonFile, Class: JsonUtil
LineNumber: 60, Type: org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method: readJsonFile, Class: JsonUtil, VariableName: File, VariableName: file, VariableName: File,
  VariableName: filePath
LineNumber: 62, Type: org.eclipse.jdt.core.dom.IfStatement, Method: readJsonFile, Class: JsonUtil
LineNumber: 67, Type: org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method: readJsonFile, Class: JsonUtil, VariableName: T, VariableName: jsonFile
LineNumber: 70, Type: org.eclipse.jdt.core.dom.ExpressionStatement, Method: readJsonFile, Class: JsonUtil
LineNumber: 131, Type: org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method: initConfig, Class: MainApp
LineNumber: 132, Type: org.eclipse.jdt.core.dom.ExpressionStatement, Method: initConfig, Class: MainApp
LineNumber: 87, Type: org.eclipse.jdt.core.dom.ExpressionStatement, Method: saveJsonFile, Class: JsonUtil
```

The logging information will be printed on the console as well as a logging.txt file will be generated, each time the code is executed.

LineNumber: 57, Type:
org.eclipse.jdt.core.dom.ExpressionStatement, Method: init,
Class: MainApp
LineNumber: 58, Type:
org.eclipse.jdt.core.dom.ExpressionStatement, Method: init,
Class: MainApp, VariableName: init
LineNumber: 79, Type:
org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method:
getApplicationParameter, Class: MainApp
LineNumber: 118, Type:
org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method:
initConfig, Class: MainApp, VariableName: Config,
VariableName: initializedConfig
LineNumber: 119, Type:
org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method:
initConfig, Class: MainApp, VariableName: String,
VariableName: configFilePathUsed
LineNumber: 121, Type:
org.eclipse.jdt.core.dom.ExpressionStatement, Method:
initConfig, Class: MainApp, VariableName: configFilePathUsed,
VariableName: Config, VariableName: DEFAULT_CONFIG_FILE
LineNumber: 123, Type: org.eclipse.jdt.core.dom.IfStatement,
Method: initConfig, Class: MainApp, VariableName:
configFilePath
LineNumber: 128, Type:
org.eclipse.jdt.core.dom.ExpressionStatement, Method:
initConfig, Class: MainApp
LineNumber: 58, Type: org.eclipse.jdt.core.dom.IfStatement,
Method: getLogger, Class: LogsCenter, VariableName: clazz
LineNumber: 44, Type:
org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method:

2) SBT

2.1) Build, run and Generate Parsing Log information using SBT build from INTELLIJ.

Once the project has been downloaded and imported as a project to INTELLIJ, kindly follow the below steps to BUILD and RUN the SBT file.
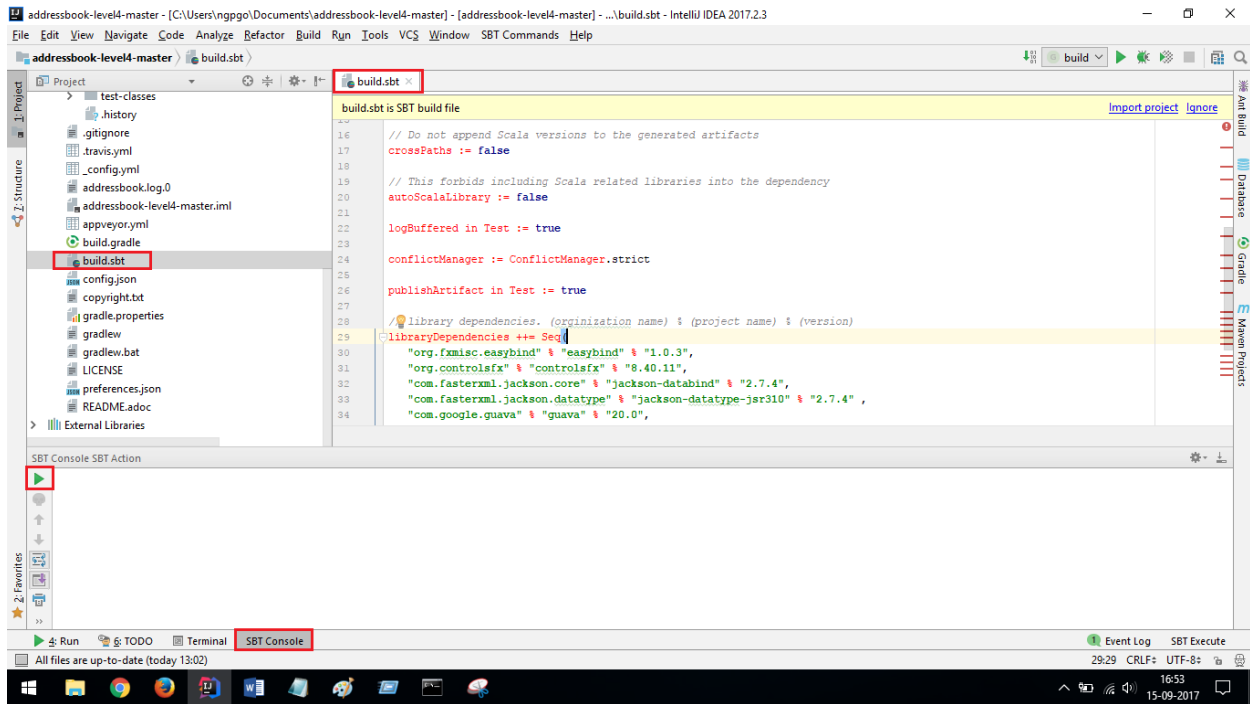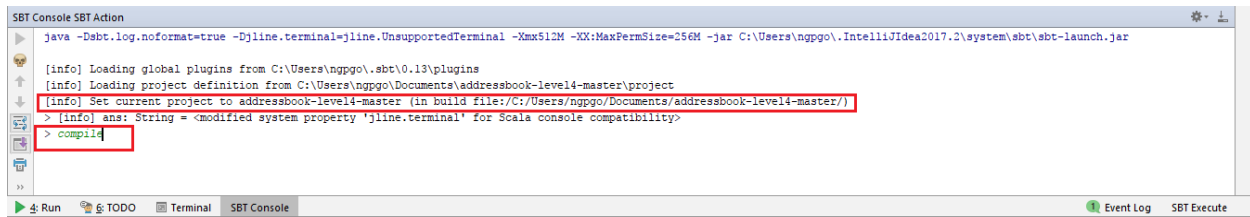
Choose the file build.sbt, where in a new window pops up. Once the window is opened click on the SBT console which can be found at the bottom of screen.

If SBT console is not found, go to view tab and under the view choose tool windows and under under tool windows choose SBT console.

Click on 'play/run' radio button to execute the SBT build
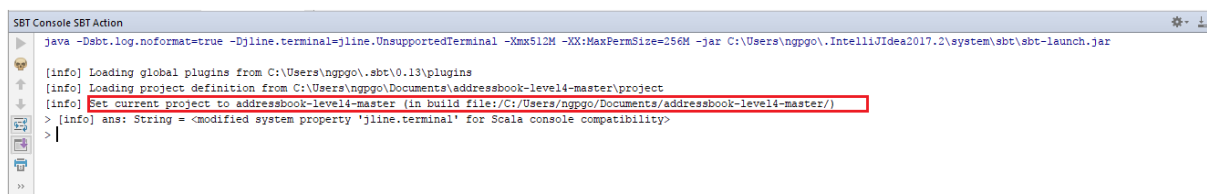
Once the current project path is set, execute the command compile
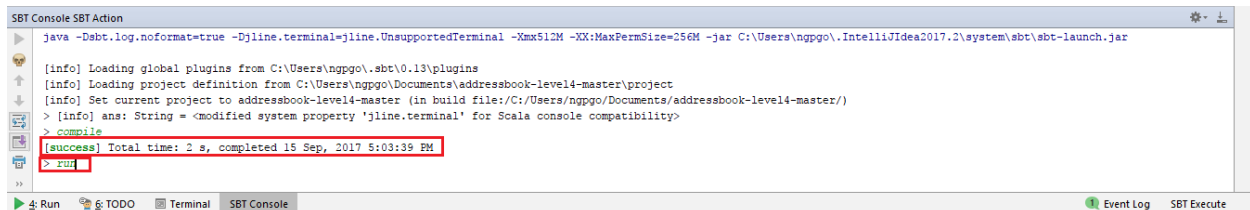
Once compilation is successful execute the command Run

The command invokes the main UI of the application and prints the parser logging information on console as well as logging.txt file.

LineNumber: 57, Type:
org.eclipse.jdt.core.dom.ExpressionStatement, Method: init,
Class: MainApp
LineNumber: 58, Type:
org.eclipse.jdt.core.dom.ExpressionStatement, Method: init,
Class: MainApp, VariableName: init
LineNumber: 79, Type:
org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method:
getApplicationParameter, Class: MainApp
LineNumber: 118, Type:
org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method:
initConfig, Class: MainApp, VariableName: Config,
VariableName: initializedConfig
LineNumber: 119, Type:
org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method:
initConfig, Class: MainApp, VariableName: String,
VariableName: configFilePathUsed
LineNumber: 121, Type:
org.eclipse.jdt.core.dom.ExpressionStatement, Method:
initConfig, Class: MainApp, VariableName: configFilePathUsed,
VariableName: Config, VariableName: DEFAULT_CONFIG_FILE
LineNumber: 123, Type: org.eclipse.jdt.core.dom.IfStatement,
Method: initConfig, Class: MainApp, VariableName:
configFilePath
LineNumber: 128, Type:
org.eclipse.jdt.core.dom.ExpressionStatement, Method:
initConfig, Class: MainApp
LineNumber: 58, Type: org.eclipse.jdt.core.dom.IfStatement,
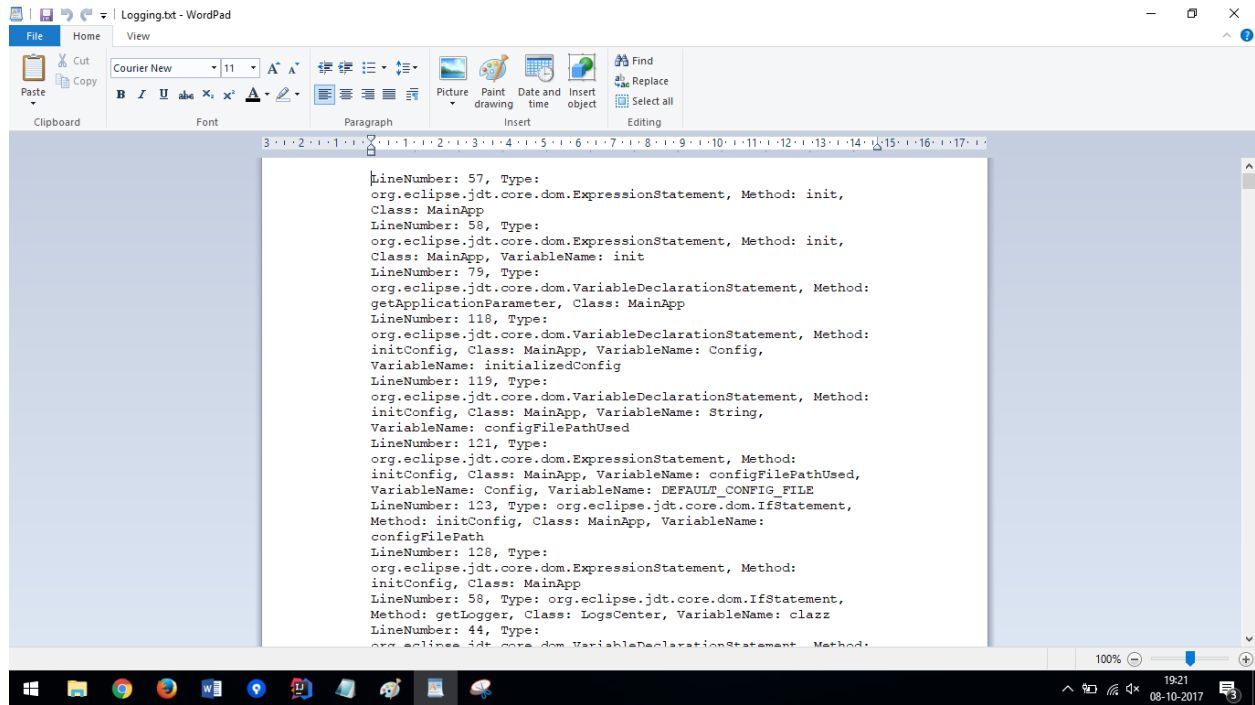Method: getLogger, Class: LogsCenter, VariableName: clazz
LineNumber: 44, Type:
org.eclipse.jdt.core.dom.VariableDeclarationStatement, Method:

3) AST Parser

The parser logic can be found in to src/main/java/ParserMain folder.