ASM Linter - User Guide

Getting Started

In order to use the ASM Java Linter to lint a Java project, you must first have built and/or compiled the project to lint in order to gain access to its bytecode .class files. These .class files will live in either a '`/target`' directory, '`/out`' directory, or a '`/build`' directory depending on what development environment and build tool is used to build the project.

For assistance in building a Java project, please refer to these tutorials depending on your needs:

- Gradle Build: https://spring.io/guides/gs/gradle/
- Maven Build: https://spring.io/guides/gs/maven/
- Intellij Build: https://www.jetbrains.com/help/idea/compiling-applications.html
- Eclipse Build: https://www.tutorialspoint.com/eclipse/eclipse_build_project.htm

User Guide

Preconfiguration (Optional)

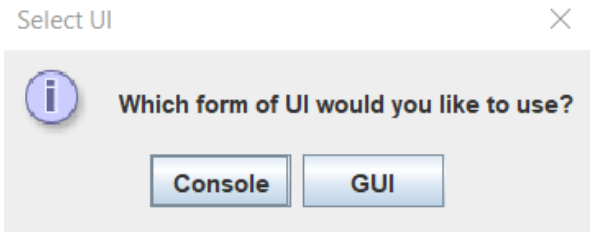
You can choose to preconfigure the linter using a `config.yaml` file to avoid configuring the linter. An example of a `config.yaml` file can be seen below.

```
directory:
  ./build/classes/java/main/ASMPracticeCode/RedundantInterface

checks:
  PoorNamingConvention: true
  EqualsHashCode: true
  HollywoodPrinciple: false
  SingletonPattern: true
  DecoratorPattern: true
  DataClass: false
  NoFinalizerCheck: false
  InformationHidingViolation: true
  SingleResponsibilityPrinciple: true
  StrategyPattern: true
  RedundantInterface: true
  ObserverPattern: false
```

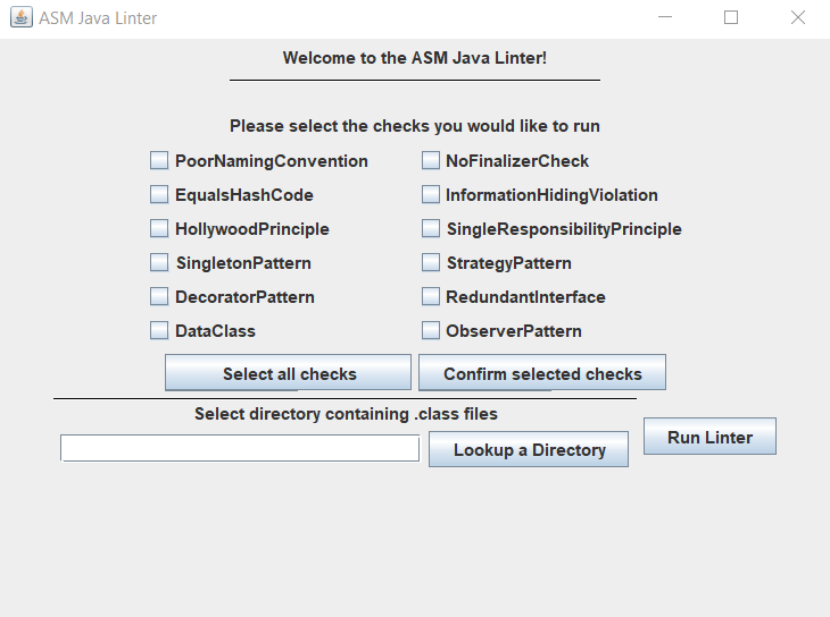
Running the Linter

Once you have built the project that you would like to lint, you can begin using the linter by running the Main class of the ASM Linter project. On startup, you will be presented with 2 options for UI: GUI and Console UI. Choose which UI version you would like to use and proceed to follow the instructions below for each version.

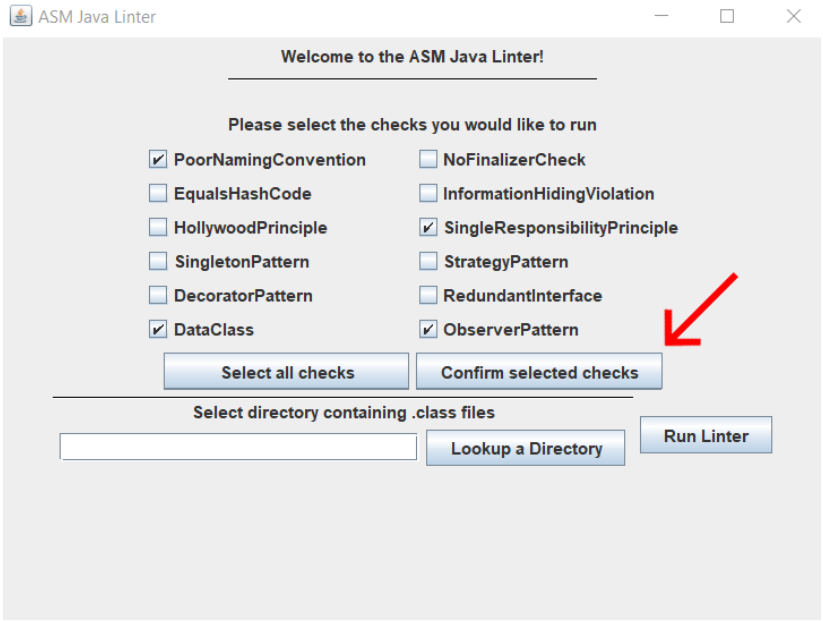


GUI

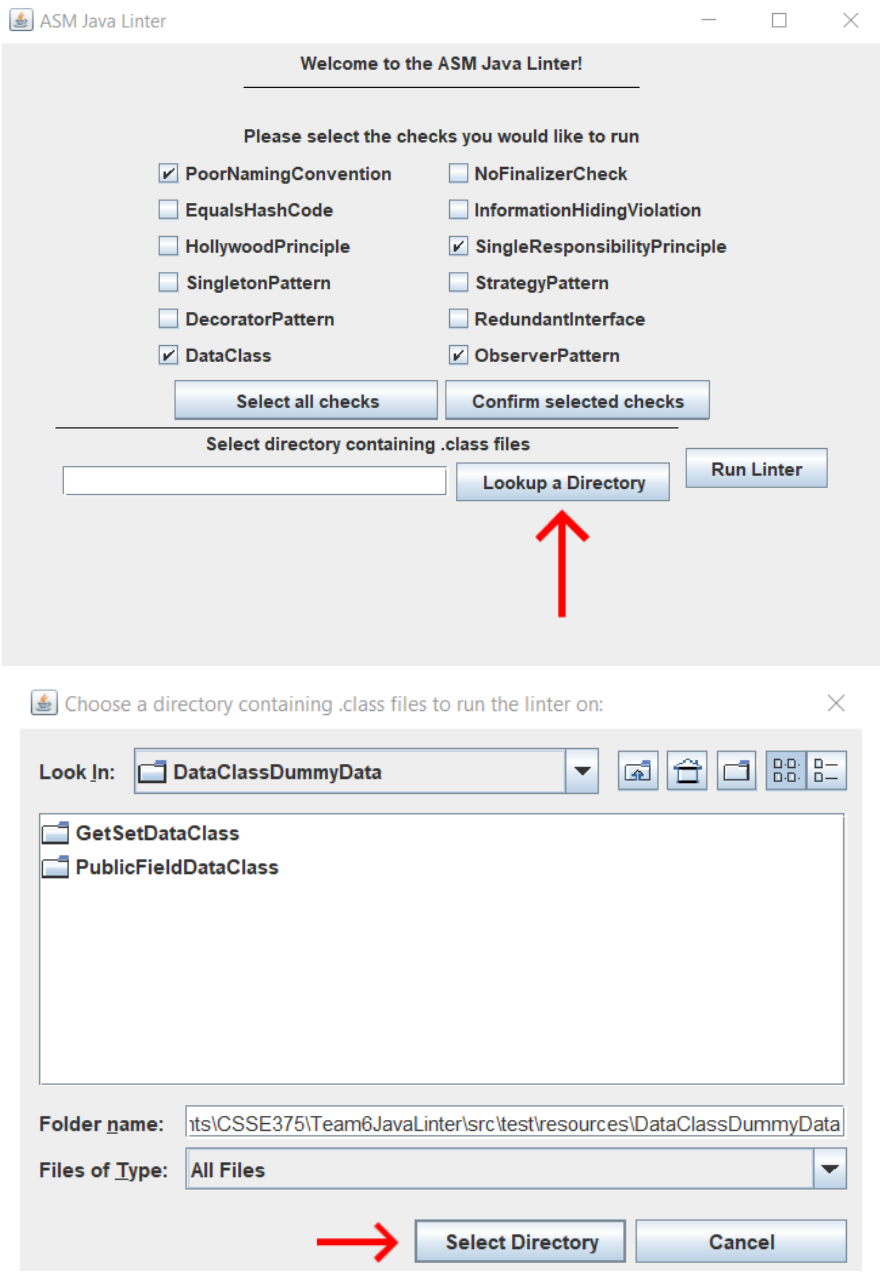
After selecting the GUI version of the UI, you will be presented with the following screen:



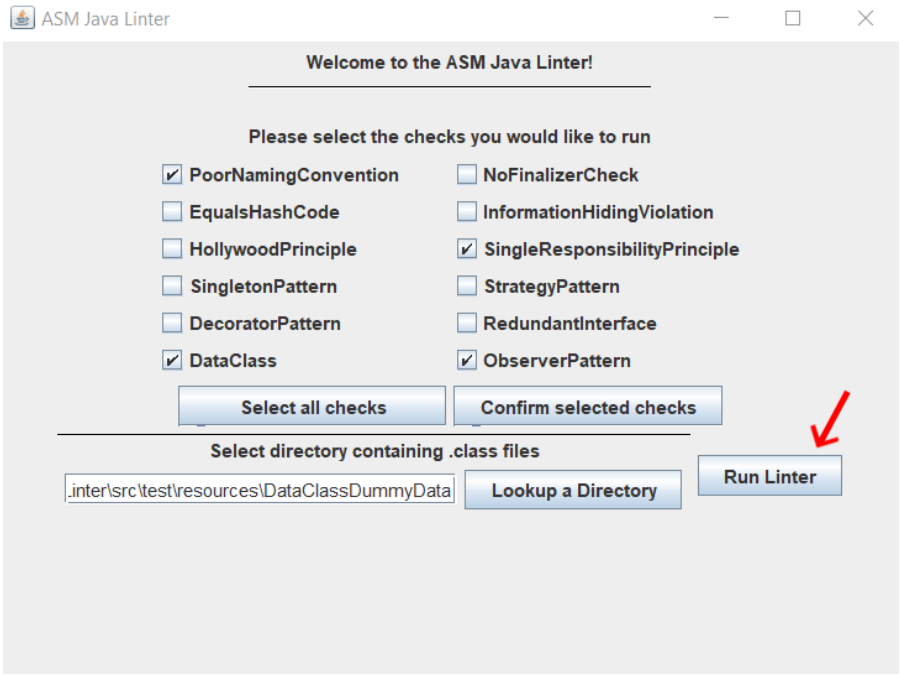
First, select the checks you would like to lint for. Be sure to click the "`Confirm selected checks`" button to ensure the linter runs those checks:



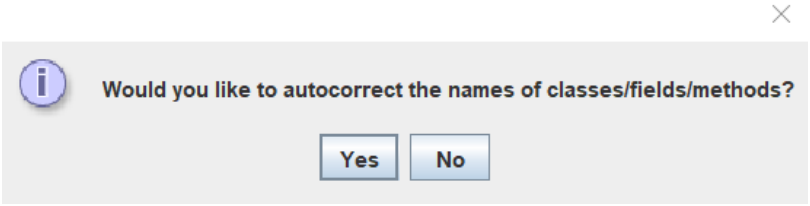
Second, either type or lookup the directory containing the `.class` files of the built project to lint:



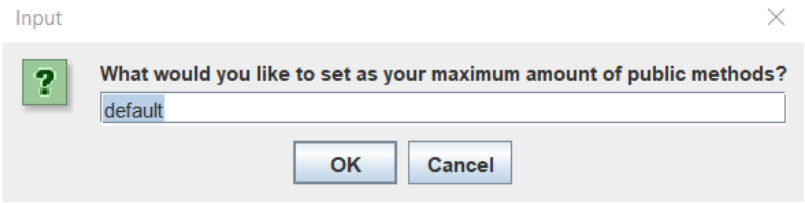
Now you can run the linter



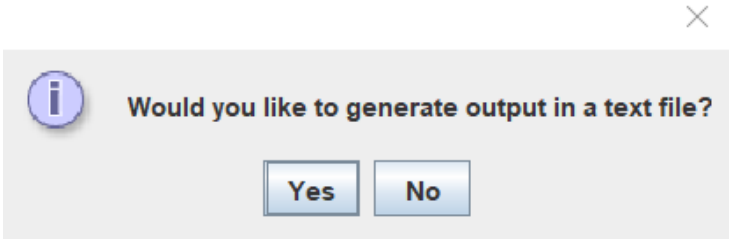
Depending on the checks selected, you may be prompted with the following options:



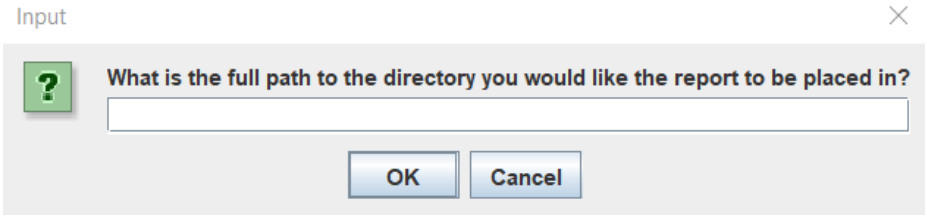
This option allows the Poor Naming Convention check to fix the names in violation



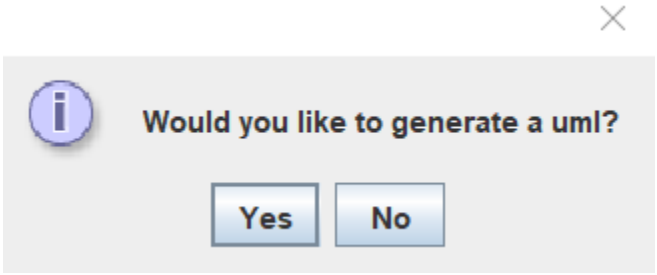
This option allows you to set a maximum amount of public methods a class can have in order to determine Single Responsibility



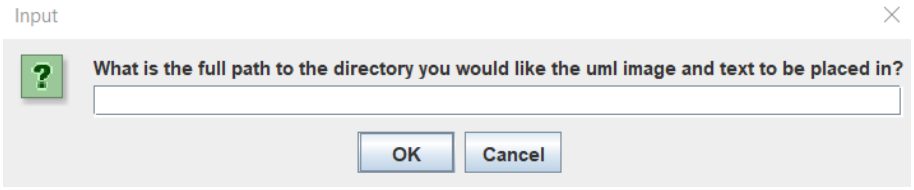
This option allows the linter to save its report in a text file in the directory of the linter for future viewing



This option allows the user to define where the output will be saved

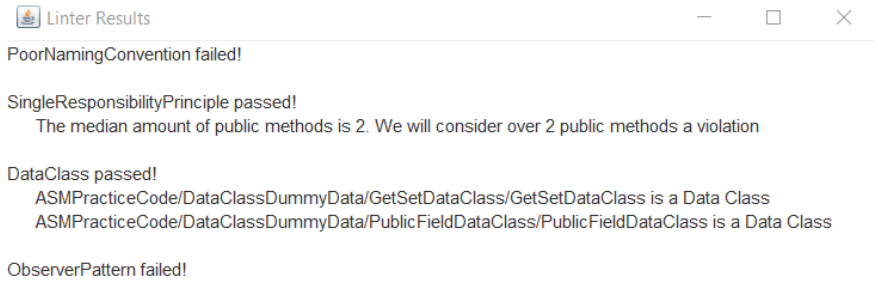


This option allows the linter to generate a uml of the linted project



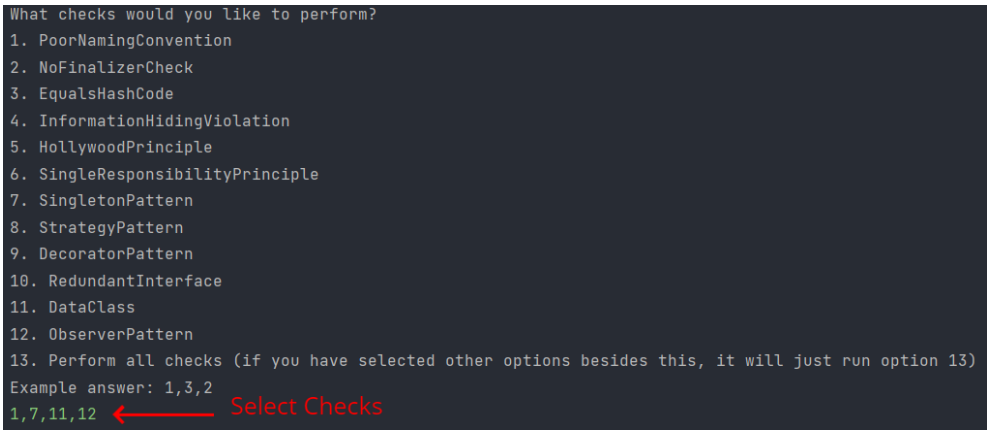
This options allows the user to define the absolute path to the directory where they would like UML to be saved

Finally, the results from the linter will appear in a new window

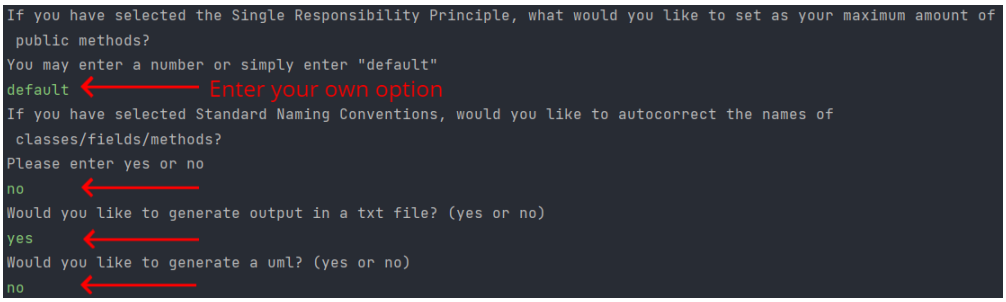


Console UI

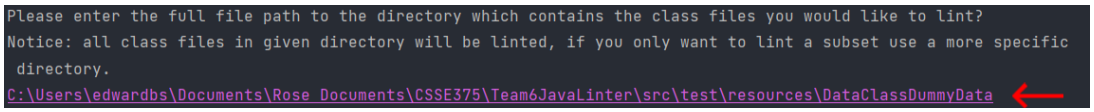
After selecting the console version of the UI, you will be presented with the following information in the console. Here you can which you can select the checks to perform:



After selecting the checks to perform, you will be presented with the following options to choose between:



After choosing your linter options, you will be prompted to provide the absolute path to the directory containing the `.class` files of the linted project:



The linter will automatically run and will output its results in the console:

```
ReportFile.txt generated in ../src/
PoorNamingConvention failed!

SingletonPattern failed!

DataClass passed!
      ASMPracticeCode/DataClassDummyData/GetSetDataClass/GetSetDataClass is a Data Class
      ASMPracticeCode/DataClassDummyData/PublicFieldDataClass/PublicFieldDataClass is a Data Class

ObserverPattern failed!
```