README.md

Binary Object loggable.Logger Implementation

Problem

Using given interface and abstract class definitions, implement a Binary Object loggable.Logger.

The basic idea is to have a binary object logger class instantiated with a file path where the log will be written. The write method accepts objects implementing a "binary loggable" interface that defines methods for writing and reading those objects to/from an array of bytes. Binary object logger will write such objects into a given binary file, and it will support reading objects back from the binary file in an iterative fashion.

You should come up with a binary file schema that supports this interface. Your implementation should be simple, robust, and with minimal overhead, so that it could be used in a performance-critical environment. This means minimal CPU and memory resources used when logger writes into the log or reads from the log. You do not have to worry about multiple threads trying to write into the same binary logger.

binarylogger.BinaryLogger interface looks as follows:

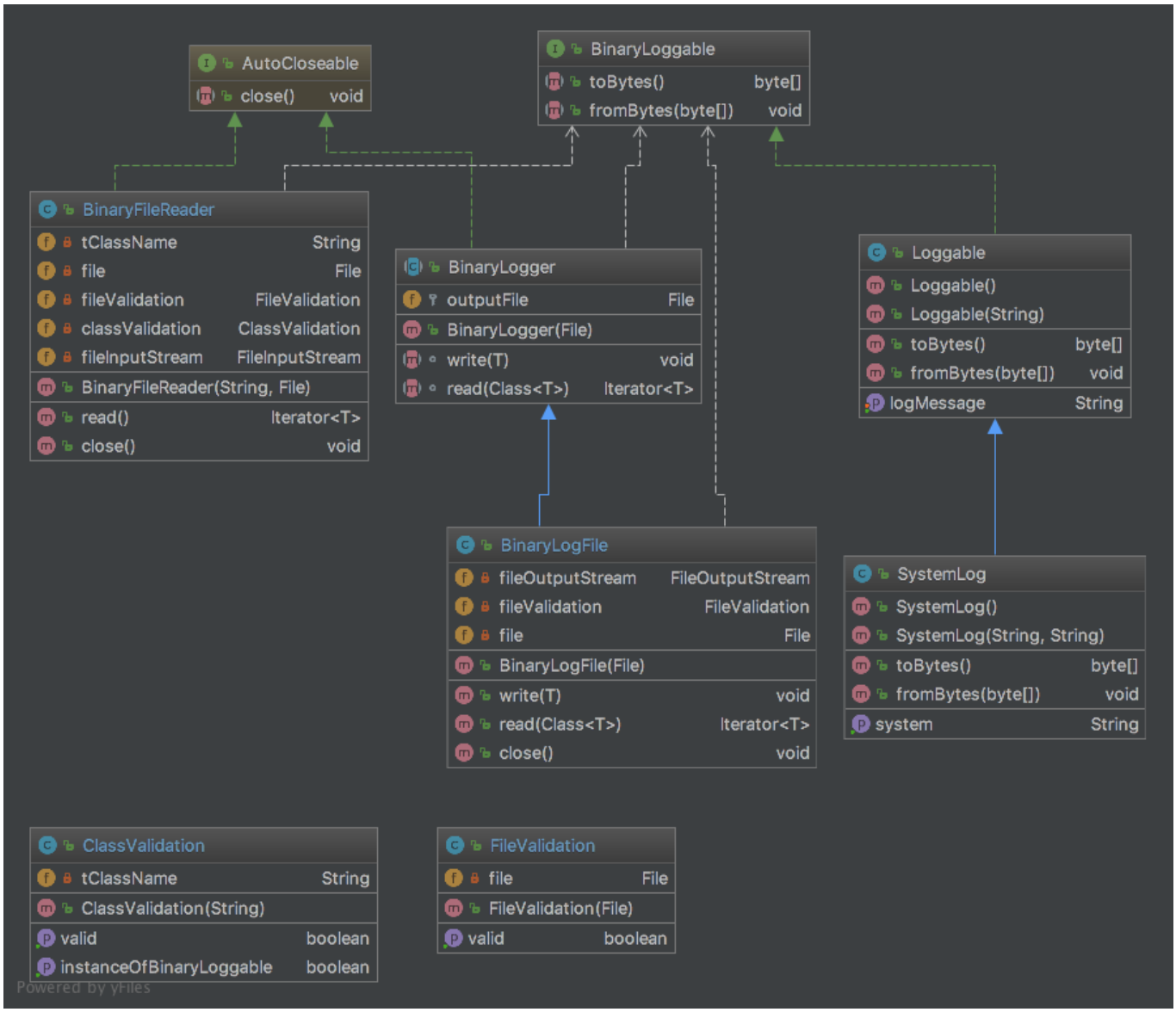
```
/**
 * binarylogger.BinaryLogger logs serialized instances of {@link BinaryLoggable} into
file.
 * It does so in such a way that it is possible to stream these
instances back
 * in an iterative fashion via the {@link #read(File, Class)} method.
 */
public abstract class binarylogger.BinaryLogger<T extends BinaryLoggable> implements
AutoCloseable {
   protected File outputFile;
   public binarylogger.BinaryLogger(File file) {
       this.outputFile = file;
}
  /**
   * Writes the serialized instance.
   *
   * @param loggable an instance of {@code loggable.BinaryLoggable} that needs to
be logged
   * @throws IOException if any IO operation fails
   */
   abstract void write(T loggable) throws IOException;
   /**
   * Read and iterate through instances persisted in the given file.
   *
   * @param clazz a class of the type T, clazz should have a public
no-arg constructor
   * @throws IOException if any IO operation fails
   */
   abstract Iterator<T> read(Class<T> clazz) throws IOException;
}
```

loggable.BinaryLoggable interface is:

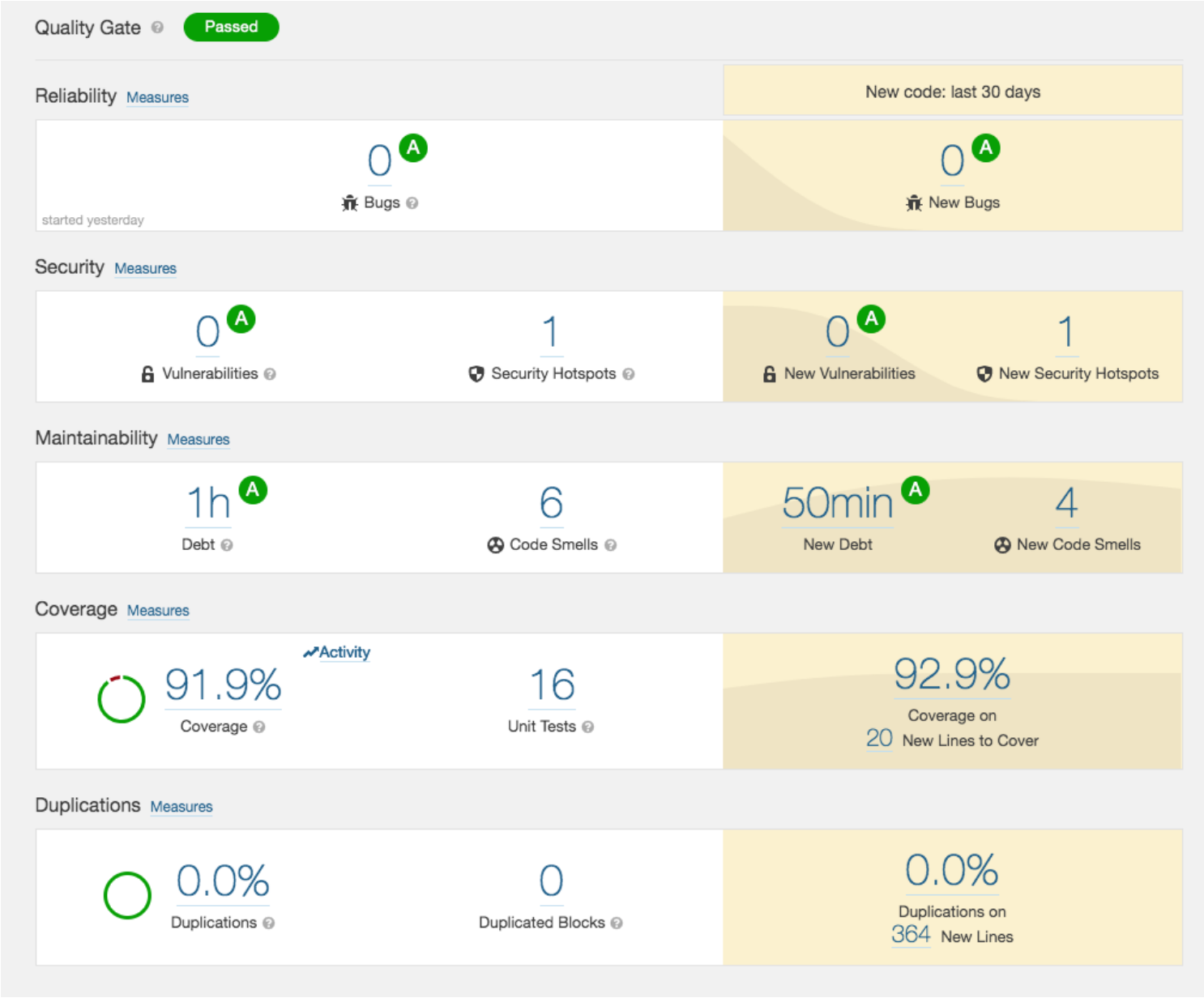
```
/**
 * loggable.BinaryLoggable represents an entity that can be logged by {@code
binarylogger.BinaryLogger}.
 */
public interface loggable.BinaryLoggable {
  /**
   * Serialize the fields of this object into a byte array.
   */
   byte[] toBytes() throws IOException;
  /**
   * Deserialize the fields of this object from given byte array.
   */
   void fromBytes(byte[] rawBytes) throws IOException;
}
```

Class Diagram

- `BinaryLogger` - This class logs serialized instances of `{@code BinaryLoggable}` into file.
- `BinaryLoggable` - This class represents an entity that can be logged by `{@code BinaryLogger}` .
- `BinaryFileReader` - This class reads `{@code BinaryLoggable}s` from a provided file.
- `BinaryLogFile` - This class is the implementation of `{@code BinaryLogger}` . It writes and reads `{@code BinaryLoggable}s` to the provided file.
- `ClassValidation` - This class checks if a class name exit and is an implementation of `{@code BinaryLoggable}` .
- `FileValidation` - This class checks the provided file exist or can be created.
- `Loggable` - This class represents an entity that can be logged by `{@code BinaryLogger}` .
- `Systemlog` - This class extends Loggable and it represents an entity that can be logged by {@code BinaryLogger}.



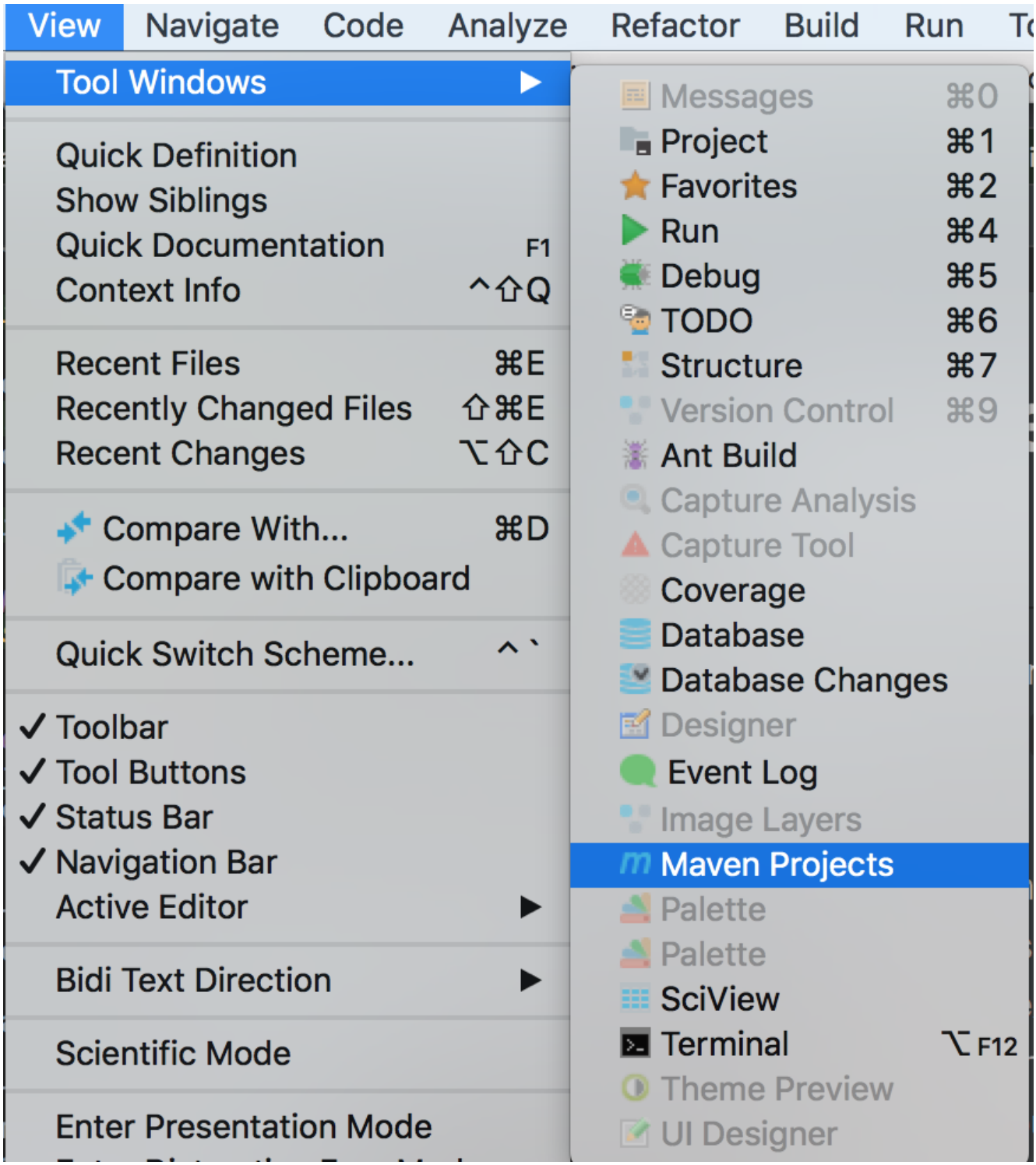
Code Analysis with Sonarcloud



Run the project from Intellij IDE

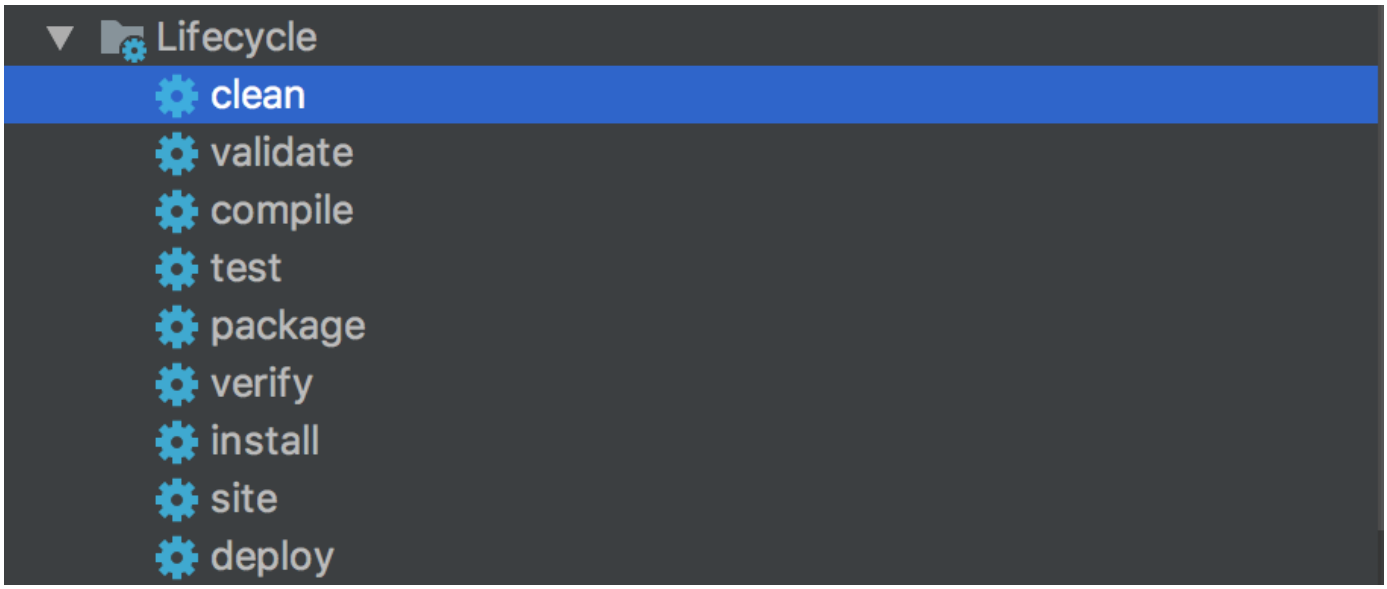
To Run the maven project from Intellij, run the Maven Project Lifecycle `clean` then `package` Maven Projects is located under `Views -> Tools Windows`

Maven Projects Location

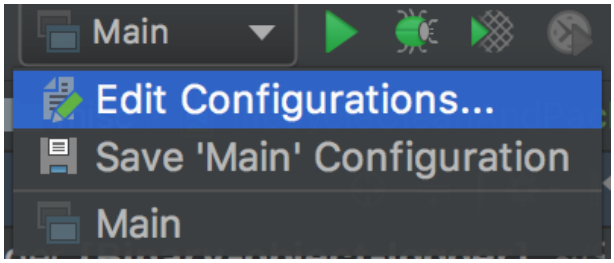


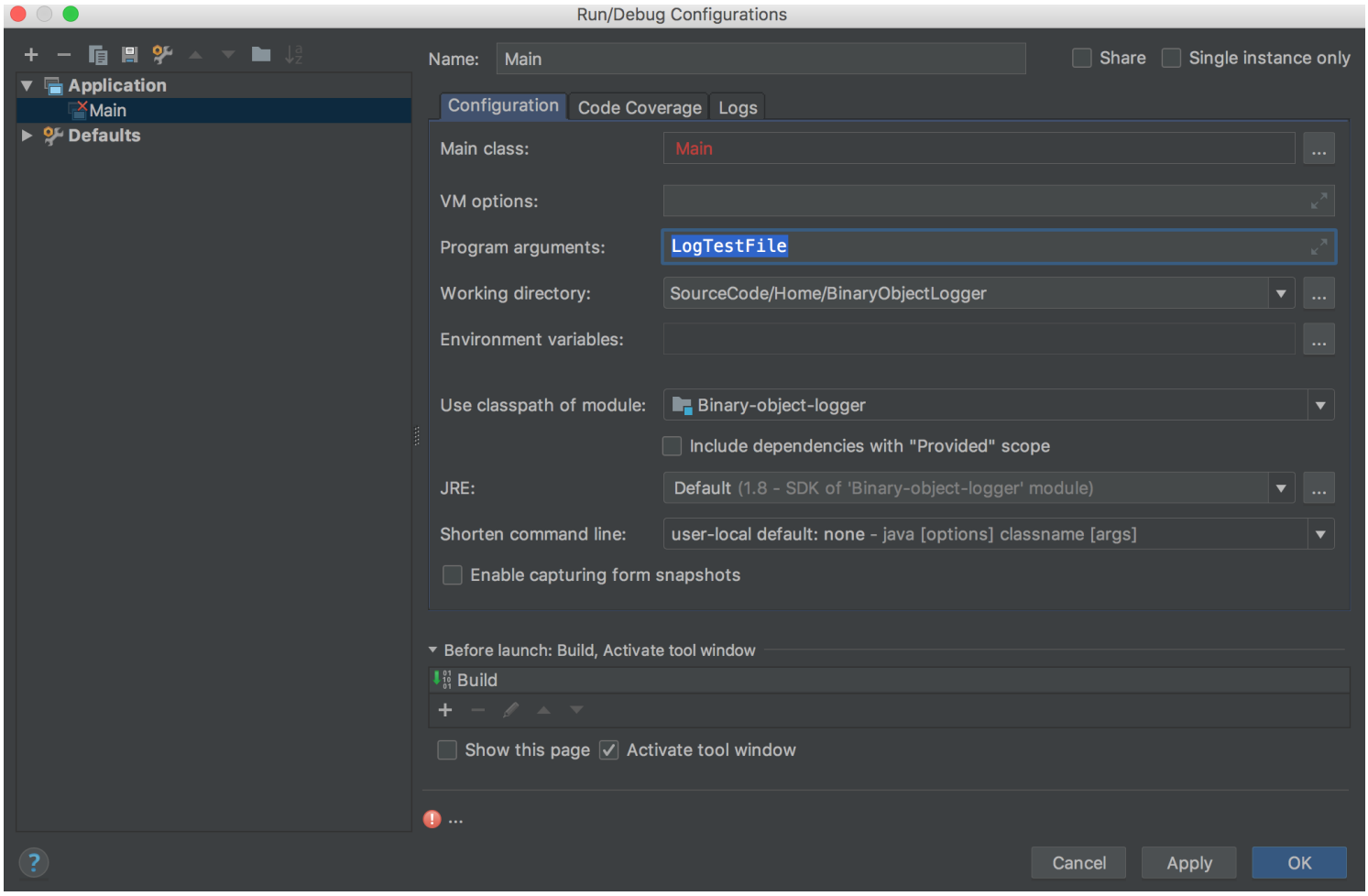
From the Lifecycle run `clean` then `pakckage` . After succeeding this step, run the main class `Main` with an argument by editing the configuration and adding the argument file `LogTestFile`

Lifecycle Clean and Package



Configuration to run the APP from Intellij IDE





Note: Feel free to use your own file!

To run the project from command line

To build and compile the project via the command line do a maven `clean` and then `package` If maven is not install on your computer use the maven wrapper with `./mvnw` instead of `mvn`

Maven Clean

```
$ mvn clean
```

Maven Response:

```
[INFO]
[INFO] ----------------< com.georgeerol:binary-object-logger >-----------------
[INFO] Building binary-object-logger 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ binary-object-logger ---
[INFO] Deleting /Users/212361198/SourceCode/Home/BinaryObjectLogger/target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.292 s
[INFO] Finished at: 2019-12-25T16:10:03-08:00
[INFO] ------------------------------------------------------------------------
```

Maven Package

```
$ mvn package
```

Maven Response:

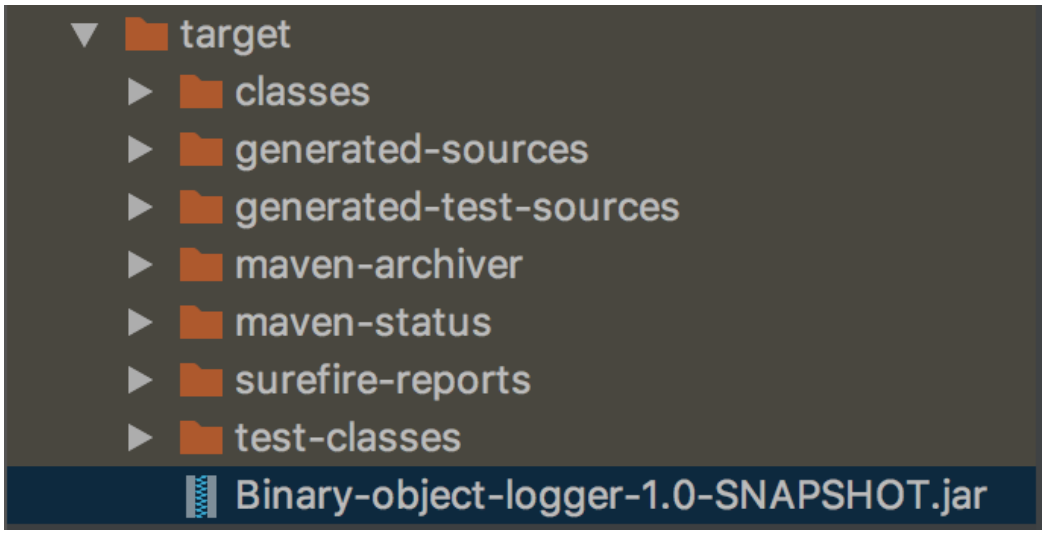
```
Tests run: 16, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:3.1.0:jar (default-jar) @ Binary-object-logger ---
[INFO] Building jar: /Users/212361198/SourceCode/Home/BinaryObjectLogger/target/Binary-object-logger-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.566 s
[INFO] Finished at: 2019-12-27T23:08:32-08:00
[INFO] ------------------------------------------------------------------------
```

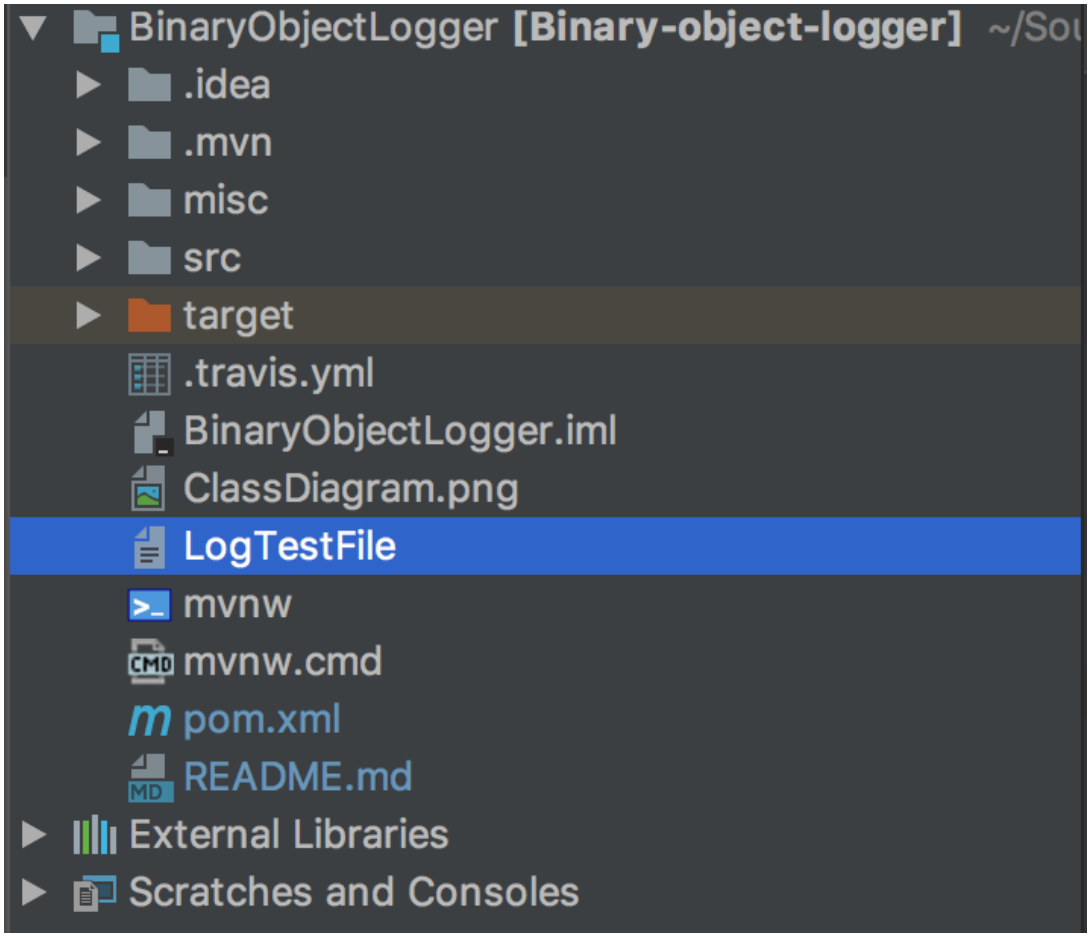
The Jar File

After running `mvn package` a target folder will be created and it will contain the jar file: `Binary-object-logger-1.0-SNAPSHOT.jar` .

Target folder



From the command line or from the IDE move the `LogTestFile` to the **target** folder



After moving the file, `cd` to the target folder and run the program via the command line:

```
$ java -jar Binary-object-logger-1.0-SNAPSHOT.jar LogTestFile
```

Note: Feel free to use your own file!