Validation & Verification

Assertion Generation Project

Gautier ROULEAU - Gwénolé LE HENAFF
Master 2 ILa 2018/2019

1. Introduction

This project was created within the ISTIC Rennes school for the V&V module.

This project is a tool that proposes new assertions to strengthen existing test cases. New assertions are based on variable values.

For our test and validation function we have used a fake project, which can be found inside our repository.

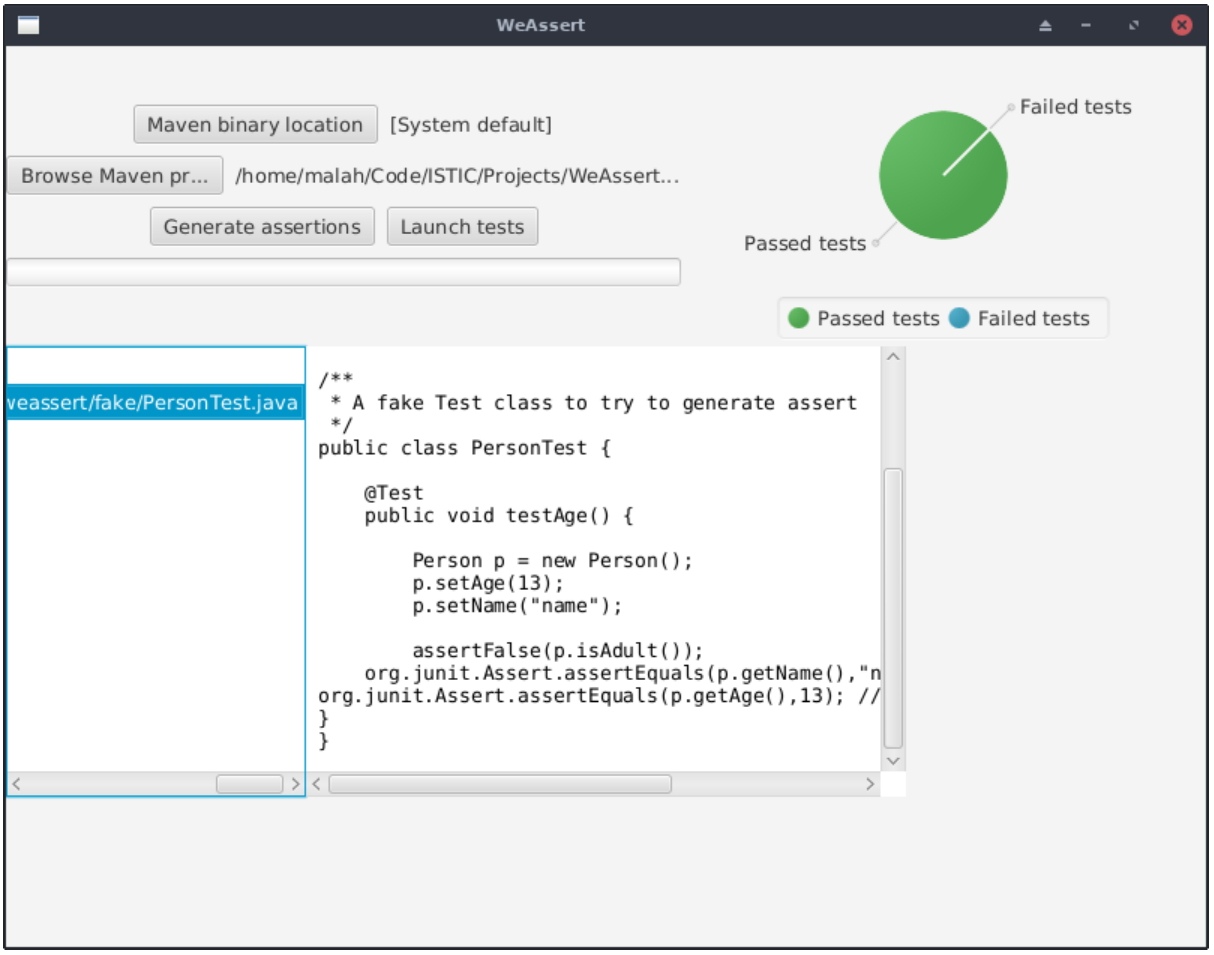
GitHub: https://github.com/ISTIC-M2-ILa-GM/WeAssert

Continuous integration: https://travis-ci.org/ISTIC-M2-ILa-GM/WeAssert

Code quality: https://sonarcloud.io/dashboard?id=fr.istic.gm.weassert

We'd like to thank Señor Oscar VERA-PEREZ for the V&V course and Madame Caroline LANDRY for the help during practical classes.

2. User Interface



The user selects the Maven executable (by default located in `/usr/bin/mvn`) and a Maven project.

After a push on the "Generate" button, the application will parse all the test classes of the project and will generate assertions for each test case.

The user can also start the tests manually to see if they pass.

The UI also allows the user to see a list of the test classes found, select one and see its source code.

3. Architecture

a. Development

Part of this project is developed using test driven development (TDD) and pair programming. We also used different design pattern likes factory, facade, adapter, and visitor.

We used code quality analysis tools to clean all of our code smells and vulnerabilities with SonarCloud, which is a cloud instance of SonarQube.

We also used test coverage with JaCoCo. We experienced some difficulties to test all methods because we used Mockito which can't interact with final classes and static methods.

b. Technology

- Language: Java 8
- Front-end: JavaFX
- Build tool: maven

We used the ASM library to parse local variables. And after experiencing some issues with the Javassist library to write a class, which can't interact with local variable, we decided to implement a tiny and hand-made source code writer.

JavaFX is a Java library which embeds a set of graphical components and containers to create a user interface using XML to describe layouts and Java to define interactions with the program.

c. Package

- GroupId: fr.istic.gm
- Artifact: we-assert

d. Build

```
$ mvn compile
```

In order for the project to work, you need to install the artifact on your local maven dependency repository before starting the app :

```
$ mvn install -DskipTests
```

Start :
Main class: fr.istic.gm.weassert.TestRunnerApp

Or with a jar:

```
$ java -jar we-assert.jar
```

The jar can be downloaded here:
https://github.com/ISTIC-M2-ILa-GM/WeAssert/releases/download/1.0.0/we-assert-compiled.zip

To generate assertions in a project, the analysed project must have all its test classes compiled with Maven and have a default target directory. You can compile them with:

```
$ mvn test -DskipTests
```

Our project can be tested with the fake project on our repository.

e. Feature done

- Variable parser
- Test analyser
- Code visitor
- Test runner
- Source code compiler
- Class loader
- Source writer
- Javassist writer
- Assertion generation
- User interface

4. Conclusion

This project allowed us to build an application with several design patterns to implement the required feature. We also learned to build an interface with JavaFX.

We learned how to use and parse byte code and source code with ASM and Javassist. This project also made us work with test driven development, mocking, and project management.

We also learned the difficulties it can be to work on this sort of project, the lack of documentation and information on how to work around the core of the code.