
Test-Generator for RxRefactor

IMPL Project: Nikolas Hanstein, Maximilian Kirschner



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1 Usage Hints

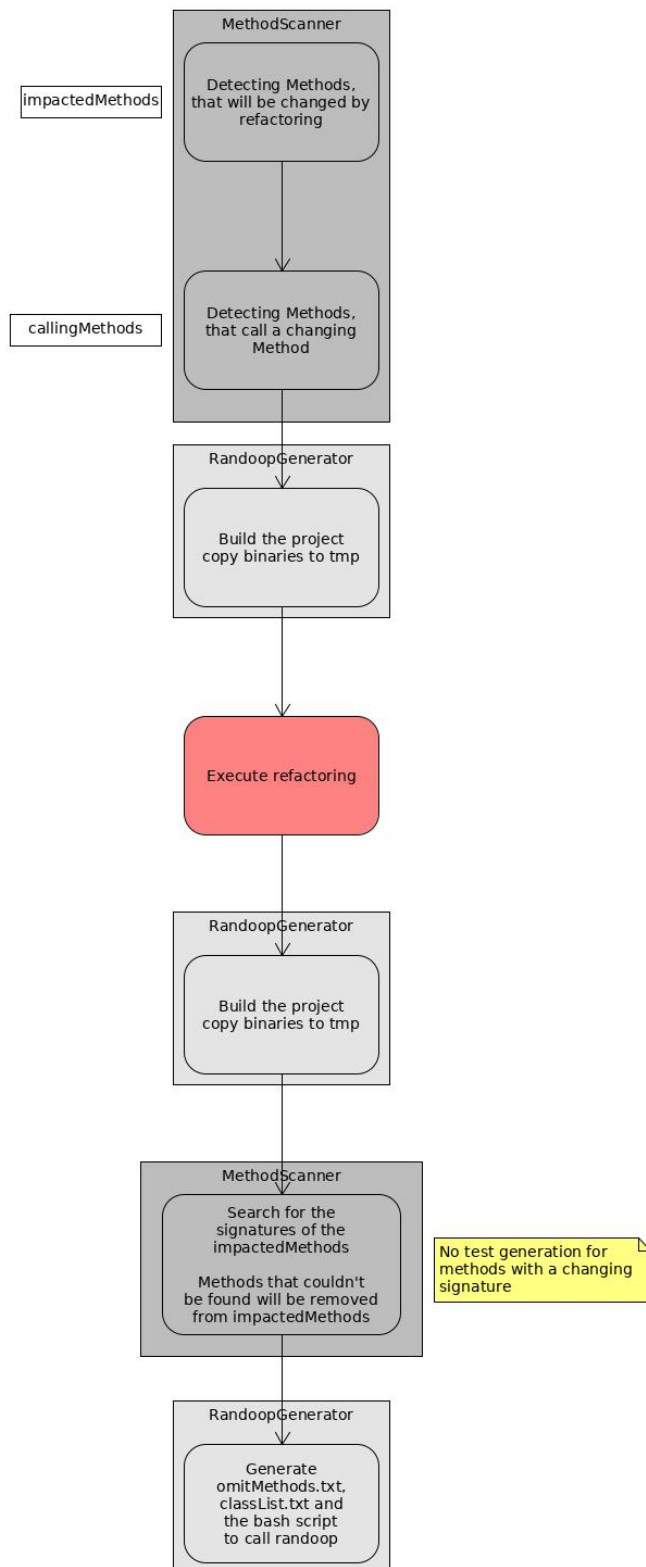
Important:

- Windows is not supported, due to some hard-coded file paths
- The tmp directory is used to save the project binaries, therefore a directory (/tmp/randoop-gen-<DATE>) is created
- Refactor and generate tests only for one project at a time, all other projects have to be closed

Steps to perform manually:

1. Use the Refactoring Tool as usual
2. Copy all project libraries, JUnit, Hamcrest, ReactiveX, Reactive-Streams and Randoop as jar-files to the a libs directory inside the created "randoop-gen" directory
3. Execute the generated bash script in the "randoop-gen" directory

2 Flow Chart



3 Code location

Our code is located in the following three packages:

- ***de.tudarmstadt.rxrefactoring.core.internal.execution.ipl***
Main Part of our code: The JavaVisitor traverses Eclipse ASTs. The MethodScanner searches for impacted and calling methods. The Randoop Generator builds the project, copies the binaries and generates the bash script
- ***de.tudarmstadt.rxrefactoring.core.internal.execution.ipl.collect***
Collections we needed, basically a Pair Class.
- ***de.tudarmstadt.rxrefactoring.core.internal.execution.ipl.filter***
FilteredArrayList which is used in the JavaVisitor to collect the nodes, that match a given Filter.

Furthermore we added some code to *de.tudarmstadt.rxrefactoring.core.internal.execution.RefactorExecution*, to obtain the ASTs, before and after refactoring. Everything that has to be executed before refactoring is added to *doRefactorProject(...)*. Everything that has to be executed after the OK-Button has been clicked is added to *run()*.

All changes in *RefactorExecution* are marked with an inline comment beginning with "IPL".