repomate-junit4 Documentation

Release 0.5.0

Simon Larsén

Mar 18, 2019

Contents:

CHAPTER 1

Install

1.1 Requirements

Important: Once you have gone through this section, you should have:

1. Repomate installed
2. A JDK (preferably 8+, 7 may work) installed
3. `junit-4.12.jar` and `hamcest-core-1.3.jar` downloaded

First of all, make sure that Repomate is installed and up-to-date. For a first-time install of Repomate, see the Repomate install docs. If you already have Repomate installed, make sure it is up-to-date.

```
python3 -m pip install --user --upgrade repomate
```

Furthermore, a JDK must be installed. `repomate-junit4` has been extensively tested with OpenJDK 8+, but should work well with JDK 7 and later. Make sure that:

1. `java` is available from the command line.
2. `javac` is available from the command line.

To be able to actually run test classes, you also need the JUnit4 and Hamcrest jars. They can be downloaded from Maven Central.

```
wget http://central.maven.org/maven2/junit/junit/4.12/junit-4.12.jar
wget http://central.maven.org/maven2/org/hamcrest/hamcrest-core/1.3/hamcrest-core-1.3.
↪jar
```

If you don't have `wget` installed, just copy the links above and download them manually.

1.2 Install plugin

To install `repomate-junit4`, simply use `pip` again.

```
python3 -m pip install --user repomate-junit4
```

Repomate should simply be able to find `repomate-junit4` if they are both installed in the same environment. To verify that it is correctly installed, run `repomate -p junit4 clone -h`. You should see some additional command line arguments added (such as `--reference-tests-dir`). See the Using existing plugins for more information on how to use plugins in general, and *Usage* for details on this plugin.

1.3 Configuration

Some options for `repomate-junit4` can be configured in the Repomate configuration file by adding the `[junit4]` section. Everything `repomate-junit4` needs to operate *can* be provided on the command line, but I strongly recommend adding the absolute paths to the `junit-4.12.jar` and `hamcrest-core-1.3.jar` files to the config file. Simply append the following to the end of the configuration file.

```
[junit4]
junit_path = /absolute/path/to/junit-4.12.jar
hamcrest_path = /absolute/path/to/hamcrest-core-1.3.jar
```

Important: All paths in the configuration file must be absolute to behave as expected.

See *CLI arguments* for a complete list of arguments that can be configured.

CHAPTER 2

Usage

2.1 Terminology and conventions

`repomate-junit4` adds some additional terminology to Repomate that you need to be familiar with to fully understand the rest of the documentation.

- **Production class:** A Java file/class in the student repo (written by the student).
- **Test class:** A Java file/class ending in `Test.java` containing tests for a namesake production class. Test classes are paired with production classes by simply appending `Test` to the production class name. For example, `LinkedList.java` would have a test class called `LinkedListTest.java`.
- **Test directory:** A directory named after a master repo, containing tests for the assignments in that repo.
- **Reference tests directory (RTD):** A directory containing test directories (as defined above).

See the *Example use case* for a more detailed look at how all of this fits together.

2.2 CLI arguments

`repomate-junit4` adds several new CLI arguments to the `repomate clone` command.

- **`-rtd|--reference-tests-dir`**
  - Path to the RTD.
  - Can be specified in the configuration file with the `reference_test_dir` option.
  - **Required** unless specified in the configuration file.
- **`-junit|--junit-path`**
  - Path to the `junit-4.12.jar` library.
  - Picked up automatically if on the `CLASSPATH` environment variable.
  - Can be specified in the configuration file with the `junit_path` option.

  - **Required** unless specified on the `CLASSPATH` variable, or in the configuration file.

- **`-ham|--hamcrest-path`**
  - Path to the `hamcrest-core-1.3.jar` library.
  - Picked up automatically if on the `CLASSPATH` environment variable.
  - Can be specified in the configuration file with the `hamcrest_path` option.
  - **Required** unless specified on the `CLASSPATH` variable, or in the configuration file.
- **`-i|--ignore-tests`**
  - A whitespace separated list of test files (e.g. `LinkedListTest.java`) to ignore.
- **`--disable-security`**
  - Disable the security policy.
- **`-v|--verbose`**
  - Display more verbose information (currently only concerns test failures).
  - Long lines are truncated.
- **`-vv|--very-verbose`**
  - Same as `-v`, but without truncation.

2.3 Example use case

Assume that we have a master repo called *fibonacci* with an assignment to implement a method that returns the n:th Fibonacci number. The master repo could then look like this:

```
fibonacci
├── README.md
└── src
    └── Fibo.java
```

To be able to test the students' implementations, we write a test class `FiboTest.java` and put it in our reference tests directory, in a test directory named after the master repository. The reference test directory would then look like this.

```
reference_tests
└── fibonacci
    └── FiboTest.java
```

Note: I strongly recommend having the reference tests in version control.

Now, assume that we have students *ham*, *spam* and *eggs*, and their student repos *ham-fibonacci*, *spam-fibonacci* and *eggs-fibonacci*. Assuming that the JUnit4 and Hamcrest jars have been configured as suggested in *Configuration*, and that the basic Repomate arguments are configured (see the Repomate config docs), we can run `repomate clone` with `repomate-junit4` activated like this:

```
$ repomate -p junit4 clone -mn fibonacci -s ham spam eggs -rtd /path/to/reference_
↪tests
[INFO] cloning into student repos ...
```

(continues on next page)

(continued from previous page)

```
[INFO] Cloned into https://some-enterprise-host/some-course-org/inda-18/ham-fibonacci
[INFO] Cloned into https://some-enterprise-host/some-course-org/inda-18/spam-fibonacci
[INFO] Cloned into https://some-enterprise-host/some-course-org/inda-18/eggs-fibonacci
[INFO] executing post clone hooks on repos
[INFO] executing post clone hooks on eggs-fibonacci
[INFO] executing post clone hooks on spam-fibonacci
[INFO] executing post clone hooks on ham-fibonacci
[INFO]
hook results for spam-fibonacci

junit4: SUCCESS
Status.SUCCESS: Test class FiboTest passed!


hook results for eggs-fibonacci


junit4: ERROR
Status.ERROR: multiple production classes found for FiboTest.java


hook results for ham-fibonacci

junit4: ERROR
Status.ERROR: Test class FiboTest failed 1 tests


[INFO] post clone hooks done
```

Note: The output is color coded when displayed in a terminal.

Let's digest what happened here. We provided the master repo name (`-mn fibonacci`) and the reference tests directory (`-rtd /path/to/reference_tests`). `repomate-junit4` then looked in the test directory matching the master repo name (i.e. *fibonacci*) test directory and found a test class `FiboTest.java`. By the naming convention, it knows that it should now look for a file called `Fibo.java` in the student repos. The following then happened when testing the repos:

- *spam-fibonacci:* The production class `Fibo.java` was found and passed the test class.
- *eggs-fibonacci:* Multiple files called `Fibo.java` were found, and `repomate-junit4` did not know which one to use. - Duplicate class names are only allowed if their fully qualified names differ (i.e. the classes are in different packages). If production code is supposed to be packaged, the test classes must also be packaged (in the same package).
- *ham-fibonacci:* The production class `Fibo.java` was found, but failed one of the tests. - Running the same command again with `-v` or `-vv` would display which test failed, and why.

Other common causes of errors include:

- No production class found for a test class.
- Compile error.
- **Security policy violation.**
  - See *Security aspects*.

This concludes the use case example, I hope you found it enlightening.

CHAPTER 3

Security aspects

There are some inconvenient security implications to running untrusted code on your own computer. `repomate-junit4` tries to limit what a student's code can do by running with a very strict JVM Security Policy. This is enforced by the Java SecurityManager. The policy used looks like this:

```
// empty grant to strip all permissions from all codebases
grant {
};

// the `junit-4.12.jar` needs this permission for introspection
grant codeBase "file:{junit4_jar_path}" {{
    permission java.lang.RuntimePermission "accessDeclaredMembers";
}};
```

This policy disallows student code from doing most illicit things, such as accessing files outside of the codebases's directory, or accessing the network. The `{junit4_jar_path}` is dynamically resolved during runtime, and will lend the actual `junit-4.12.jar` archive that is used to run the test classes sufficient permissions to do so.

This policy seems to work well for introductory courses in Java, but there may be snags because of how restrictive it is. If you find that some permission should definitely be added, please open an issue about it. There are plans to add the ability to specify a custom security policy, but currently, your only choice is to either use this default policy or disable it with *–disable-security*.

Important: The security policy relies on the correctness of the Java SecurityManager. It is probably not bulletproof, so if you have strict security requirements, you should only run this plugin inside of a properly secured environment (for example, a virtual machine).

CHAPTER 4

repomate-junit4 Module Reference

This module reference is intended for developers contributing to repomate-junit4 and should not be considered a stable API.

4.1 junit4

Plugin that runs JUnit4 on test classes and corresponding production classes.

Important: Requires javac and java to be installed, and having hamcrest-core-1.3.jar and junit-4.12.jar on the local macine.

This plugin performs a fairly complicated tasks of running test classes from pre-specified reference tests on production classes that are dynamically discovered in student repositories. See the README for more details.

class repomate_junit4.junit4.**JUnit4Hooks**

act_on_cloned_repo(*path*)

Look for production classes in the student repo corresponding to test classes in the reference tests directory.

Assumes that all test classes end in Test.java and that there is a directory with the same name as the master repo in the reference tests directory.

Parameters **path** (Union[str, Path]) – Path to the student repo.

Return type HookResult

Returns a plug.HookResult specifying the outcome.

clone_parser_hook(*clone_parser*)

Add reference_tests_dir argument to parser.

Parameters **clone_parser** (ConfigParser) – The clone subparser.

Return type None

config_hook(*config_parser*)
Look for hamcrest and junit paths in the config, and get the classpath.

Parameters **config** – the config parser after config has been read.

Return type None

parse_args(*args*)
Get command line arguments.

Parameters **args** (Namespace) – The full namespace returned by

:param argparse.ArgumentParser.parse_args():

Return type None

4.2 _java

Utility functions for activities related to Java.

This module contains utility functions dealing with Java-specific behavior, such as parsing package statements from Java files and determining if a class is abstract.

repomate_junit4._java.**extract_package**(*class_*)
Return the name package of the class. An empty string denotes the default package.

Return type str

repomate_junit4._java.**extract_package_root**(*class_*, *package*)
Return the package root, given that **class_** is the path to a .java file. If the package is the default package (empty string), simply return a copy of **class_**.

Raise if the directory structure doesn't correspond to the package statement.

Return type Path

repomate_junit4._java.**fqn**(*package_name*, *class_name*)
Return the fully qualified name (Java style) of the class.

Parameters

- **package_name** (str) – Name of the package. The default package should be an empty string.
- **class_name** (str) – Canonical name of the class.

Return type str

Returns The fully qualified name of the class.

repomate_junit4._java.**generate_classpath**(**paths*, *classpath=''*)
Return a classpath including all of the paths provided. Always appends the current working directory to the end.

Parameters

- **paths** (Path) – One or more paths to add to the classpath.
- **classpath** (str) – An initial classpath to append to.

Return type str

Returns a formated classpath to be used with java and javac

repomate_junit4._java.**is_abstract_class**(*class_*)
Check if the file is an abstract class.

Parameters **class** – Path to a Java class file.

Return type bool

Returns True if the class is abstract.

repomate_junit4._java.**javac**(*java_files*, *classpath*)
Run javac on all of the specified files, assuming that they are all .java files.

Parameters

- **java_files** (Iterable[Union[str, Path]]) – paths to .java files.
- **classpath** (str) – The classpath to set.

Return type Tuple[str, str]

Returns (status, msg), where status is e.g. Status.ERROR and the message describes the outcome in plain text.

repomate_junit4._java.**pairwise_compile**(*test_classes*, *java_files*, *classpath*)
Compile test classes with their associated production classes.

For each test class:

1. Find the associated production class among the java_files

2. Compile the test class together with all of the .java files in the associated production class' directory.

Parameters

- **test_classes** (List[Path]) – A list of paths to test classes.
- **java_files** (List[Path]) – A list of paths to java files from the student repo.
- **classpath** (str) – A base classpath to use.

Return type Tuple[List[HookResult], List[HookResult]]

Returns A tuple of lists of HookResults on the form (succeeded, failed)

repomate_junit4._java.**properly_packaged**(*path*, *package*)
Check if the path ends in a directory structure that corresponds to the package.

Parameters

- **path** (Path) – Path to a Java file.
- **package** (str) – The name of a Java package.

Return type bool

Returns True iff the directory structure corresponds to the package name.

4.3 _junit4_runner

repomate_junit4._junit4_runner.**get_num_failed**(*test_output*)
Get the amount of failed tests from the error output of JUnit4.

Return type int

repomate_junit4._junit4_runner.**parse_failed_tests**(*test_output*)
Return a list of test failure descriptions, excluding stack traces.

Return type str

repomate_junit4._junit4_runner.**run_test_class**(*test_class*, *prod_class*, *classpath*, *verbose=False*, *security_policy=None*)

Run a single test class on a single production class.

Parameters

- **test_class** (Path) – Path to a Java test class.
- **prod_class** (Path) – Path to a Java production class.
- **classpath** (str) – A classpath to use in the tests.
- **verbose** (bool) – Whether to output more failure information.

Return type Tuple[str, str]

Returns ()

repomate_junit4._junit4_runner.**security_policy**(*classpath*, *active*)

Yield the path to the default security policy file if active, else yield None. The policy file is deleted once the context is exited.

TODO: Make it possible to use a custom security policy here.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index

Symbols

A

C

E

F

G

I

J

P

R

S