Assignment 2: Estimating Testing Effectiveness

Start Assignment

Due Oct 19 by 11:59pm **Points** 6 **Submitting** a text entry box or a file upload
Available Oct 10 at 12am - Oct 19 at 11:59pm

Goals for This Assignment

By the time you have completed this work, you should be able to:

- Measure code coverage
- Perform mutation testing
- Write tests that maximize code coverage and killed mutants

This assignment may be completed in groups.

Step-by-Step Instructions

Step 1: Download and Unzip Needed Code

Download the code here (https://kyledewey.github.io/comp587-fall18/lecture/week_6/coverage_mutation_template.zip), and unzip it. This is the same code that we have been working with in class.

Step 2: Install Maven

Install Maven (https://maven.apache.org/). If you're using Linux, binary packages are usually available through your distribution's package manager. If you're using Mac OSX, binary packages are available via Homebrew (https://brew.sh/).

Step 3: Compile Code with Maven

Code is already provided in `src/main/java/trees/TreeOperations.java`. This code should compile as-is. You should be able to compile your code with the following command:

```
mvn compile
```

If your code doesn't compile, it may be because your version of Java is improperly set in the provided `pom.xml`. To check this, run the following:

```
java -version
```

This will give you output like the following:

```
openjdk version "1.8.0_171"
OpenJDK Runtime Environment (build 1.8.0_171-b10)
OpenJDK 64-Bit Server VM (build 25.171-b10, mixed mode)
```

The version number in the first line should match up with the version number in the `maven.compiler.source` and `maven.compiler.target` lines of `pom.xml`. In this case, the version number of `1.10` is *incorrectly* set in `pom.xml`, and should instead be set to `1.8`.

Step 4: Write Implementation

As with the in-class exercise, you should write the following three methods in `src/main/java/trees/TreeOperations.java`:

1. A breadth-first search over a tree (https://en.wikipedia.org/wiki/Breadth-first_search), returning the items in the tree. Different method signatures are possible for this. If you're stuck, you can use the following signature:

```
        public static <A> ArrayList<A> bfs(final Node<A> root)
```

2. A pre-order traversal over a tree (https://en.wikipedia.org/wiki/Tree_traversal#Pre-order_(NLR)), returning the items in the tree. Different method signatures are again possible; you can use the following if you're stuck:

```
            public static <A> ArrayList<A> preorder(final Node<A> root)
```

3. A calculation of the maximum depth of a given tree. Different method signatures are again possible; you can use the following if you're stuck:

```
            public static <A> int maxDepth(final Node<A> root)
```

You may write helper methods if you wish.

Step 5: Write Tests

Write JUnit (https://junit.org/junit5/) tests for your methods in `src/test/java/trees/TreeOperationsTest.java`. Try to write tests that you think will cover all parts of your code. You can run your tests with the following command:

```
mvn test
```

All of the tests you write should pass.

Step 6: Measure Coverage and Possibly Write More Tests

You can measure the coverage of your tests with the following commands:

```
mvn test
mvn jacoco:report
```

These comments will compute the coverage information. You can then view the results by opening `target/site/jacoco/index.html`. The results have links that will allow you to explore your code, and see line and branch coverage information.

For the purposes of this assignment, you should have at least **90%** line and branch coverage over your entire implementation (line and branch coverage are separate measurements). If you have less than 90% coverage, you should return to step 5 and incrementally improve your coverage. The **only** exception is if you can explain *why* the coverage is below 90%, **and** exactly why it cannot be improved.

Step 7: Perform Mutation Testing and Possibly Write More Tests

You can perform mutation testing (also known as mutation analysis) with the following commands:

```
mvn test
mvn org.pitest:pitest-maven:mutationCoverage
```

These commands will perform mutation testing. You can view the results by opening `target/site/pit-reports/<<representation of most recent date and time>>/index.html`. This will show you the mutants that were created, along with which were killed and which survived.

For the purposes of this assignment, you should kill at least **90%** of the mutants. If you have less than 90% killed mutants, revisit step 5 and improve this ratio. The **only** exception is if you can explain *why* this ratio is below 90%, **and** exactly why it cannot be improved.

Step 8: Submit Everything

Zip up everything in your `coverage_mutation_template` directory, *including* the `target` directory. **Be sure your `target` directory holds up-to-date coverage and mutation testing results.** Name your zipfile `coverage_mutation_solution`, and submit it on **Canvas**. In the comments for the submission, list everyone you worked with, if applicable. If you had less than 90% coverage or mutants killed, similarly explain in the submission comments on Canvas.