SE 333 Software Testing

Assignment 2: Calculating metrics

Due date: September 27, 2021, 11:59pm
Late penalties: 1% late penalty per day for up to 7 days. No late submission will be accepted after 7 days.

Overview

This assignment helps you practice both the calculation of the reliability metrics and also introduces junit unit tests. Some unit tests already exist in the project template. Your code should pass these tests without altering them. You will also add some of your own unit tests.

The template project provides 2 classes:

- MetricsCalculator
- MetricsCalculatorTest

Part 1

Initially, all the tests in MetricsCalculatorTests will be failing. This is because MetricsCalculator contains only stub implementations of the required functions:

```
public double getMTBF(int[] events, int scope);

public double getMTTR(int[] events, int scope);

public double getAvailability(int[] events, int scope);

public double getAffectedUsers(int sessionsPerHour, int sessionLength, double
downTime);

public double getReliability(int sessionsPerHour, int sessionLength, double
downTime, int scope);
```

For part 1 of the assignment, you add code to these functions so that they correctly implement the indicated metric. Doing so should make the tests pass.

Your functions should verify that the provided inputs are sensible, for example an array of downtime event durations cannot be empty or null. Give some thought to what other constraints make sense. When invalid values are given as parameters, throw IllegalArgumentException with an appropriate error message.

See comments in the source code for details on each function.

Tips:

1. getReliability should use getAffectedUsers
2. a simple way to sum up an array is
   ```
   int aSum = Arrays.stream(arrayOfInts).sum();
   ```

Part 2

The existing unit tests only test 2 happy path scenarios for each function. In this part of the assignment, you need to add tests for failures. Most if not all of these tests will use assertThrows(...) as its main verification but other tests are acceptable, unless those tests would be better implemented using assertThrows, for example, do not use try/catch in your tests.

Delivery

The deliverable for this assignment is a zip file containing your updated project, named:

<user>-metrics.zip

Where <user> is replaced with your Depaul user name. For example, my delivery would be DWALKE30-metrics.zip

NOTE: the only acceptable way to create this zip file is using the gradle script provided with the project. Running the script produces the zip file in the build/dist directory within the project. Generate and upload this zip file. If you have difficulty generating this file, get help. **Hand crafted zip files will not be accepted.**

Grading criteria

1. Each of the 5 production methods are implemented correctly (5 points)
2. MetricsCalculator covers probable error conditions (2 points)
3. MetricsCalculatorTest covers the exceptions thrown by MetricsCalculator (3 points)
4. Coding following class style guidelines (1 point)
5. zip file was created correctly, using gradle build process (1 point)