# SE 333 Software Testing

# Assignment 5 Black Box Testing

Due date: May 4, 11:59 pm

> Late penalties: 1% late penalty for each day of lateness, up to 7 days. No work later than 7 days will be accepted.

## The Objective

The objective of this assignment is to design and implement test suites using Equivalence partitioning and Boundary value analysis.

## Overview

The template for this assignment contains a jar file but no source code for the classes under test. Your assignment is to develop solutions to this assignment, using the specifications as your only inputs as to how the applications work.

You must not:

1. Decompile the jar file
2. Re-implement the classes defined in the jar file

## Program Shipping Cost

### Specification

The program calculates the total amount of each customer order for an on-line retailer. The total amount of an order is the sum of the total cost of the purchased items, the shipping cost, and applicable state sales tax.

- A customer may choose from *Standard Shipping* or *Next-Day Shipping*. The cost of Standard Shipping is $10 per order. The cost of Next-Day Shipping is $25 per order.
- Shipping is only available to destinations in the US.
- Customers who are residents of IL, CA, and NY will be charged 6% state sales tax, excluding the shipping cost.
- The Standard Shipping charge will be waived if the total purchase (excluding sales tax) is over $50.

## Function signature

```
public static double calculateTotal(
        double rawTotal,
        Orders.ShippingMethod shippingMethod,
        String destinationState)
```

**Shipping methods**

```
public static enum ShippingMethod {
    Standard,
    NextDay;
```

### Problems

1. Identify the equivalence classes for each input variable, including both valid and invalid equivalence classes.

2. Using parameterized tests and other Junit techniques, design and implements a test suite with fewest test cases possible that meets the requirements of *weak normal* test.

3. Extend the test suite in 3 with fewest additional test cases possible to stratify the *weak robustness* test.

4. Design and implement a test suite with fewest test cases possible that meets the requirements of *strong normal* test.

5. Extend the test suite in 4 with fewest additional test cases possible to stratify the *strong robustness* test.

6. Design and implement additional tests for the boundary values related to rawTotal

### Tips

1. You may find it useful to work out the test cases in a spreadsheet or similar document. Remember that a Normal test case would include an expected numeric result from the test, so a spreadsheet would help you arrive at the right value. Turn in any such documents with your code.

2. The code you are testing contains defects. Do not write a test that lets the test pass, if you must violate the specifications to do so. When in doubt, assume the specification are right, not the code.

3. The FunctionalTests examples from a previous week shows how to use a CSV file as input to parameterized tests, as well as other examples that might be useful to you.

**Submission**

1. Zip up your solution using maven in the usual way
2. Upload the zip as well as any spreadsheets or other supporting documents you created for the assignment.  Supporting documents should be separate files, not added to the maven zip file.