CS139 Algorithm Development
Activity 06A: Testing Methods

Objectives

At the end of this activity, students will be able to:

- Review and use method terminology
- Given a method header and its description, define appropriate test cases
- Use team members in different roles.

Part 1 – Warm-up – Applying terminology

Given the temperature program below, answer the questions that follow.

```
/**
 * TemperatureConverter contains methods to convert temperatures.
 * @author Nancy Harris
 * @version 09/28/2012
 */
public class TemperatureConverter
{
    /**
     * main will exercise the two conversion methods
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        System.out.println("100 C s/b 212F is " + convertToFahrenheit(100.0));
        System.out.println("32 F s/b 0C is " + convertToCelcius(32.0));
    }
    /** convertToFahrenheit converts a celcius temperature to
     *  Fahrenheit equivalent
     * @param celcius The celcius temperature to convert
     * @return the equivalent Fahrenheit temperature
     */
    public static double convertToFahrenheit(double celcius)
    {
        double fahrenheit;

        fahrenheit = (celcius * 9.0 / 5.0) + 32;

        return fahrenheit;
    }
    /** convertToFahrenheit converts a celcius temperature to
     *  Fahrenheit equivalent
     * @param fahrenheit The Fahrenheit temperature to convert
     * @return the equivalent celcius temperature
     */
    public static double convertToCelcius(double fahrenheit)
    {
        double celcius;

        celcius = (fahrenheit - 32) * 5.0 / 9.0;

        return celcius;
    }
}
```

1. **Individually**, fill in the chart below.

On which line will you find a method header?		What is the name of the method?	
On which line will you find a parameter?		What is the name of the parameter?	
On which line will you find a return statement?		What is being returned?	
On which line will you find a call to a method in this file?		What is the name of the method?	
On which line will you find an argument?		What is the argument?	

2. **Group**, compare answers.

Part 2 – Thinking about testing

A value returning method has a set of one or more inputs and for each combination of inputs, an expected output. Becoming a programmer means learning how to test your code such that your program reliably produces the expected output.

Given a method with the following method documentation and header:

```
/**
 * Given a String parameter, return a new String with the
 * characters in the reverse order.
 * For example, given abc the method would return cba.
 *
 * @param text The text to reverse
 * @return The reversed String
 */
public static String reverse(String text)
```

3. In your group, consider at least 4 **different** test cases that you would use to test your code. Write them in the chart below:

Argument value	Result value

4. Looking at your test cases, what properties makes each case different? In other words, except for the particular letters or symbols used, what makes one case different from another? In the chart below, indicate for each value from the chart above what makes that case different from the one before it.

Argument value	What are you testing for?

5. **BOARD – Recorder, write the Argument and Result values on the board for your group. Presenter, be prepared to explain how your test cases are different from one another.**

6. On a separate piece of paper, assuming that reverse is in the same class as main, write the body of main that would print out the result of each test case showing the argument, the expected value and then the value obtained by the method call.

Part 3– A more complicated example

A value returning method has a set of one or more inputs and for each combination of inputs, an expected output. Becoming a programmer means learning how to test your code such that your program reliably produces the expected output.

Given a method with the following method documentation and header:

```
/**
 * Produce the amount of tax on the given amount based on the following rules:
 *   If the taxType is 'X' or 'x' (exempt), then tax amount is 0.
 *   If the taxType is 'M' or 'm', then tax is 11% of the amount.
 *   If the taxType is 'F' or 'f', then tax is 2% of the amount.
 *   If the taxType is anything else, then tax is 5% of the amount.
 *
 * @param amount The amount of the sale
 * @param taxType A code for the type of items were purchased
 * @return The amount of tax
 */
public static double calculateTax(double amount, char taxType)
```

7. **BOARD** – What is the minimum number of tests required to test each of the different cases?

8. What are those cases and what are you testing for? (There may be more lines than you need)

Argument value	Expected value	What are you testing for?

9. On the back of this page, write a main method to carry out 3 of the test cases you have defined above.

This space provided for your main methods.