Week 2 – Test Plan Assignment

For the Chapter 2 autograded programming projects, the Revel autograder expects a class named **Exercise**. If you create a class with a different name in an IDE, remember to change the name to **Exercise** when submitting to the Revel autograder.

Instructions:

- Download and edit this document to fill in your test plans.
- Apply the techniques described in the lesson "Week 2 Specification-based Testing" for the following tasks:
    - Testing Task #1: Average.java.
    - Testing Task #2: ViralSpread.java.
    - Testing Task #3: Chapter 2 Programming Project 1.
    - Testing Task #4: Chapter 2 Programming Project 5.
- When editing test cases, fill in the expected I/O (input/output) **prior** to running the program. Fill in the actual I/O and status (Pass or Fail) after running the program.
    - Display the input values in bold. It's ok to wrap a line of output across multiple lines if the table cell is not wide enough to display it on one line. Assume input is entered on the same line as the prompt, even if it is displayed on a separate line in the test case.
    - Refer to the Revel sample runs for exact input/output expectations per project.
- Describe the lessons learned while implementing each assigned programming project. What challenges did you encounter and how did you solve them?

Note: This submission is only for the test plans. The Java code for your programming project solutions should be submitted in the Revel environment for grading.

Testing Task #1 – Average.java

TODO: Decide how to provide Average.java

Click here to download a copy of Average.java.

Click here to run Average.java. Create your own editable version by clicking "Fork".

Run Average.java for each test case listed below. Fill in the actual output and status (Pass/Fail).

	Expected I/O	Actual I/O	Status
1	Enter 2 numbers: **10 24** Average = 17.0		
2	Enter 2 numbers: **13 42** Average = 27.5		

There is a calculation error in the code. Identify and fix the error so the actual and expected output match. Rerun the tests to confirm your solution is correct.

	Expected I/O	Actual I/O	Status
1	Enter 2 numbers: **10 24** Average = 17.0		
2	Enter 2 numbers: **13 42** Average = 27.5		

Include a screen print of your code solution:

Testing Task #2 – ViralSpread.java

TODO: Decide how to provide ViralSpread.java

Click here to download a copy of ViralSpread.java.

Click here to run ViralSpread.java. Create your own editable version by clicking "Fork".

R0 (R-naught) is a term to describe the reproduction rate of infectious pathogens. R0 of 2 implies an infected person may infect 2 others. Three iterations result in $2^3 = 8$ infections. In comparison, an R0 of 5 results in $5^3 = 125$ infections.

The program reads in a value for R0 and calculates the number of infections after 3 iterations, i.e. $(R0)^3$, but there is an error in the code. Refer to https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#pow-double-double-

Run the program for each test case listed below. Fill in the actual output and status (Pass/Fail).

	Expected I/O	Actual I/O	Status	Comment
1	Enter R0: **2** 3 iterations result in 8 infections			R0 = 2
2	Enter R0: **3** 3 iterations result in 27 infections			R0 = 3
3	Enter R0: **5** 3 iterations result in 125 infections			R0 = 5

Identify and fix the error in the program. Rerun the tests to confirm your solution is correct.

	Expected I/O	Actual I/O	Status	Comment
1	Enter R0: **2** 3 iterations result in 8 infections			R0 = 2
2	Enter R0: **3** 3 iterations result in 27 infections			R0 = 3
3	Enter R0: **5** 3 iterations result in 125 infections			R0 = 5

Include a screen print of your code solution:

Testing Task #3 : Chapter 2 Project 1 Test Plan

- Fill in the expected I/O for test cases 3 and 4 based on the values specified in the comment. Define two additional test cases 5 and 6, describing your choice of input values in the comment.
- Run your Project 1 code solution for each test case and fill in the actual I/O and status based on the program execution.
- Try to fix errors identified by the tests.
- If you are unable to get a test case to pass, mention it in the lessons learned.

	Expected I/O	Actual I/O	Status	Comment
1	Enter the subtotal and a gratuity rate: **10 15** The gratuity is $1.5 and total is $11.5			Sample run subtotal $10 rate 15%
2	Enter the subtotal and a gratuity rate: **12.8 10.5** The gratuity is $1.344 and total is $14.144			subtotal $12.8 rate 10.5%
3				subtotal $0 rate 20%
4				subtotal $27.50 rate 0%
5				
6				

Lessons learned while implementing programming project 1:

Testing Task #4 : Chapter 2 Project 5 Test Plan

- You should be able to predict the expected I/O for test case #3 based on the values from tests #1 and #2.
- Come up with input values for test cases 5 – 7, using a positive number for any variable not assigned to 0 in the comment. Fill in the expected I/O for test cases 5-7 prior to running the program.
- Run your Project 5 code solution for each test case and fill in the actual I/O and status based on the program execution.
- Try to fix errors identified by the tests.
- If you are unable to get a test case to pass, mention it in the lessons learned.

	Expected I/O	Actual I/O	Status	Comment
1	Enter investment amount: **1000.56** Enter annual interest rate in percentage: **4.25** Enter number of years: **1** Accumulated value is $1043.92			Sample run. $1000.56 investment, 4.25%, 1 year
2	Enter investment amount: **1043.92** Enter annual interest rate in percentage: **4.25** Enter number of years: **1** Accumulated value is $1089.16			$1043.92 investment, 4.25%, 1 year
3				$1000.56 investment, 4.25%, 2 years
4	Enter investment amount: **2000** Enter annual interest rate in percentage: **5** Enter number of years: **10** Accumulated value is $3294.01			$2000 investment, 5%, 10 years
5				0 years
6				0 rate
7				0 investment

Lessons learned while implementing programming project 5: