Learning Objectives:

- Practice jUnit Tests
- Unit test for exceptions
- Use online resources to understand the domain you are testing for

Description:

- Write jUnit tests for the following methods in class Vector (provided):
  - `componentAt`
  - `plus`
  - `minus`
  - `times`
  - `distanceTo`

In order to write unit tests for a method we need to understand its required functionality.
To learn more about how to add, subtract, and multiply vectors check out:
http://www.mathsisfun.com/algebra/vectors.html

To learn more about the Eulcidean distance check out
http://en.wikipedia.org/wiki/Euclidean_distance

Here is a check-list that should help you with your jUnit tests:

- ☐ Make sure to use descriptive names (for variables, methods, and class names)
- ☐ When you write a test for methodX that methodX needs to be called to find out the actual value
- ☐ The expected value should be as simple as possible (literals are ideal)
- ☐ Each test case needs to include an assertion to validate the functionality of the method
  http://www.tutorialspoint.com/junit/junit_using_assertion.htm
- ☐ The assertion should be based on the specification (the required behavior) of the method not the implementation
- ☐ Whenever you need to call another method to validate the behavior of a given methodX that other method should to be unit tested before you use it in the test for methodX.

Many times you need multiple jUnit tests to test a single method
For each method that throws an exception write a separate jUnit test that verifies that the right Exception is thrown
In this video I demonstrate how to use multiple unit tests to test for a given method and how to test for exceptions:
https://www.youtube.com/watch?v=I2aQlEsmAlY

Turning in:

Create a zip file that includes **ONLY the jUnit test code** (TestVector.java)
I will run it against my own Vector class.
Turn it in via Canvas