- For lab 6 we will be creating a JUnit 5 test for all of the methods of program 4
- Most of the tests are reasonbly adequate, except for the tests for isConnected() and getConnectedSet(), and that is due to the lack of comprehensive sample graphs
- For the lab my preference is that everyone uses eclipse to create the JUnit 5 tests, but I am OK with doing it outside of eclipse with compiling and running the test on the commandline
- What needs to be completed for the lab
  - At a minimum, a shell of all of the required Graph class methods
  - At a minimum, all of the JUnit tests implemented in a JUnit 5 test file
    - I have provided code for some of the JUnit tests
  - At least one test passing
    - The isDirected() tests should be really easy to get passing
- Once you have the minimum working, continue working on your Graph class until close to the end of the activity section

- If you haven't started your graph class yet
  - I posted a video of the below, should take around 12 minutes, including adding one test
  - Create the class
  - Add the three fields
    - ArrayList<Integer> vertices
    - ArrayList<Edge> edges
    - boolean isDirected
    - Mine are private and final
  - Add the one parameter constructor, assigning the boolean parameter to isDirected
    - And instantiating vertices and edges
  - Add the no parameter constructor, having it call the one parameter constructor with a parameter of false
  - Add
    - public boolean isDirected returning the value of isDirected
    - public int getNumberOfVerrtices() returning the size of vertices
    - public int getNumberOfEdges() returning the size of edges

- If you haven't started your graph class yet (cont)
  - Add (cont)
    - public String toString() returning “”
    - public void addVertex(int v) throws GraphException returning nothing
    - public void addEdge(int from, int to) throws GraphException returning nothing
    - public void isConnected() returning false
    - public java.util.HashMap<Integer> get ConnectedSet() returning “new java.util.HashMap<Integer>();”
- At this point you can start adding the JUnit tests
  - It took me about 10 minutes to get to this point if I didn't have any of the graph class started
- Now if you add the JUnit test, with tests for addEdge() and addVertex() you can execute the JUnit tests
  - Both tests fail, but you have something to work with now
- If you add the graph declaration, mine is “`garrison_Graph g;`” prior to the @BeforeAll and @BeforeEach
- And add the graph instantiation in @BeforeEach, mine is “`g = new garrison_Graph();`”
- You should be able to implement the three versions of the isDirected test pretty quickly and already have three tests working

- The lab seems more like a typing lab than a programming lab
  - But the dream for me is that as you add the tests, you are gaining some understanding as to what they are accompishing
- The desire is that we all understand
  - Why we are creating each of the 19 tests
  - What each of the 19 tests is testing
  - How each test is accomplishing what it is testing
  - Once we have all 19 tests implelemted, we should have a pretty comprehensive set of test for most of program 4

- For the lab, see “testing and junit test” lecture notes
  - Slides 5 – 7 tells you how to get started in eclipse
  - Slides 8 – 34 goes over all of the tests that we are interested in, there are 19 (I have 21 since I kept both copies of the third duplicate edge test and something else that I haven't identified)
  - I have posted the following
    - The lines of code that define the String for the sample graphs and the expected results for getConnectedSet() for the sample not connect undirected graph
    - The testGetNumberOfEdges() code
      - You should be able to copy and paste this for the testAddEdge() test and then make some minor modifications
    - The testIsConnectedUndirectedAndConnected() & testIsConnectedUndirectedAndNotConnected() code
      - You should be able to copy and paste these for the other two isConnected() tests, and them make some minor modifications
    - The testAddVertex() code
      - You should be able to copy and paste this for the testGetNumberOfVertices() and then make some minor modifications
    - The testGraphExceptionForDuplicateVertex() code
      - You should copy and paste it for the other two exception tests and then make the appropriate modifications
    - The testGetConnectedSetDirectedAndNotConnected() & testGetConnectedSetUndirectedAndNotConnected() code
      - You should copy and paste these for the two getConnectedSet() tests

- You should end up with the following 19 tests

test	purpose	slide
testAddVertex()	tests addVertex() is adding vertices and not adding duplicate vertices for an undirected graph, this test is provided	16
testAddEdge()	tests addEdge() is adding edges and not adding duplicate edges for an undirected graph	18
testToString()	tests toString() for an undirected graph	34
testGetNumberOfVertices()	tests getNumberOfVertices() for an undirected graph	12
testGetNumberOfEdges()	tests getNumberOfEdges() for an undirect graph, this test is provided	14
testGraphExceptionForDuplicateVertex()	tests addVertex() throws GraphException when a duplicate vertex is attempted to be added for an undirected graph, this test is provided	20
testGraphExceptionForDuplicateEdge()	tests addEdge() throws GraphException when a duplicate edge is attempted to be added for an undirected graph	21
testGraphExceptionForDuplicateEdge2()	tests addEdge() throws GraphException when a duplicate edge is attempted to be added for an undirected graph	22
testGraphExceptionForInvalidEdge()	tests addEdge() throws GraphException when an invalid edge is attempted to be added for asn undirected graph	23
testGraphExceptionForDuplicateEdge3b()	tests addEdge() does not throw a GraphException when an edge with reversed vertices is added to a directed graph	25
testIsDirectedForUndirectedGraph()	tests that the 0 parameter constructor instantiates an undirected graph	9
testIsDirectedForDirectedGraph()	tests that the 1 parameter constructor with a parameter of true instantiates a directed graph	10
testIsDirectedForUndirectedGraph2()	tests that the 1 parameter constructor with a parameter of false instantiates an undirected graph	10
testIsConnectedUndirectedAndConnected()	tests that isConnected() returns true for sample_undirected_graph_1.txt, this test is provided	28
testIsConnectedUndirectedAndNotConnected()	tests that isConnected() returns false for sample_undirected_graph_2.txt, this test is provided	29
testIsConnecteDirectedAndConnected()	tests that isConnected() returns true for sample_directed_graph_1.txt	30
testIsConnecteDirectedAndNotConnected()	tests that isConnected() returns false for sample_directed_graph_2.txt	31
testGetConnectedSetDirectedAndNotConnected()	tests that getConnectedSet(2) returns the correct connected subset for sample_directed_graph_2.txt	32
testGetConnectedSetUndirectedAndNotConnected()	tests that getConnectedSet(2) returns the correct connected subset for sample_undirected_graph_2.txt	33

- Getting credit for the lab
  - Demo to a TA/CA that your JUnit5 tests run and at least one of the tests passes
  - e-mail me a copy of the Junit5 test file with a subject of “cs 140 lab 6”