**Nicholas Skinner**

**CS362**

**Midterm**

**A**)You are to test a method that takes as input name of a student and a list of grades, as follows: public void **printNameGrade(String name, List<Integer> grade).** The specifications state the following:

* The method reads the grades in the list and writes the student name and the highest grade that in the list.
* For example, if the name was Ali Aburas and the Grades were 40, 90,85, 50, and 60 the output would be:
  + Ali Aburas 90
* The student name is to be alphabetic characters 3 to 15 characters in length.
* Each grade may be a value in the range of 0 to 100, whole numbers only.
* A maximum of five grades may be entered for each student.

How will you test such a method**? (Not unit tests, just verbal explanation of how you test).** Try to cover as many scenarios as possible. Keep the description of your scenarios as short as possible**. (25 points)**

----------------------------------------------------------------------------------------------------------

**1A) How will you test such a method?**

There are a number of ways to test this method, an important part of this however, could be input validation. If one student's name is longer or shorter than the maximum and minimum, some sort of error could be thrown. Another portion of input validation would be to make sure that the 1 to 100 value for their grade does not have an invalid character, and if it does, we will need to display an error. An additional case could be attempting to enter more or less total grades for the individual than the projected maximum of five, what will it display if the student has more? What will it display if they have none? Valid test cases are important as well, making sure that a correct case is able to pass through to the next portion of the method. After all of these cases, we can begin testing permutations of each to ensure that the methods work in their proper ways.

After input validation, another way to test these methods will be the individual calls and cases they are expected to be in, possess, and output through the process. This would include any manipulation of a variable, and any replacing or casting involved in a process, such as iterating through the grades to find the highest, we may want to call checks to make sure that in every case it is correctly assigning the highest value to be the highest.

The final way I would approach testing these scenarios is to make sure the output, and the printed string match, and that the printed string is what you would expect it to be, the printed string could also be involved in checking for names that are longer than a certain length, and a highest value that may be outside of the 1 to 100 grade range.

**B**)David needs to test a method that adds two integer numbers “**public int add(int num1**, **int num2)**”. Provide good input partitioning (Equivalence Partitioning) to David so that David will be able to use a varying set of inputs to test the method. Explain each choice that you make in few words and provide with an explanation some input values for each partition. (**25 points**)

----------------------------------------------------------------------------------------------------------------

An equivalence test is a test that is meant to be representative of a potentially infinite number of test cases, and this specific scenario is requiring that said values be ints. Ints have a very shallow number of potential results, as they are only able to add up to be ints. For these cases I will be focusing on the following tests to test this program:

Scenario 1:

num1 = valid int

num2 = valid int

In this case we have how the program is expected to work, which is always important to double check. We want to ensure that the numbers reach the correct and desired output, and that there are no snags along the way

Scenario 2:

num1 =valid int

num2 = invalid int

An invalid int in this scenario can be a number of things, but I’ve settled on the Invalid Int being a string in this situation, a string somehow, we will want to make sure that it does not add these values, and that it does not spit out anything other than invalid inputs, or some sort of error, as if the inputs are not validated, something can go wrong in the program. The inverse scenario of num2 being valid and num1 being invalid will be tested as well.

Scenario 3:

num1=valid

num2=null

This is done in the interest of ensuring that if a value is left in a state of null, that it will act with the method in a meaningful way. While null is technically not a number, it is occasionally able to be passed into a method in a non number state. This will ensure that adding either does, or does not work with NaN characters, and also that our inputs are validated in some form, unless we get an undesirable result. The inverse will also be tested

Scenario 4:

num1=null

num2=invalid int

This scenario exists to be a catch all for the remaining situations, there will be permutations of invalid and invalid, null and null, and every combination of those inputs. This exists to ensure that values that are not wanted in the loop will not end up in it. This will show that the method is able to disallow bad inputs and will not use them if they are given.