ITC205 AA Reflective Task.

Name: Corie Rhodes

Student ID: 11774079

Date: 11/12/2022

Please answer the following questions in a few sentences in the context of your experience with the subject and the AA assignment.

1. **Explain what constitutes a good test in the context of unit and integration testing.**

A good integration test is one that effectively evaluates the integration of different modules or components of a system to ensure they work together as expected. Several key characteristics make a good integration test:

- Relevance: A good integration test focuses on the interfaces between different components or modules of the system and verifies that they are working together as expected. The test should be designed to expose any issues or defects that may arise from the integration of these components.
- Completeness: A good integration test should cover a wide range of scenarios and test cases, including both positive and negative test cases. This helps to ensure that the system is thoroughly tested and that all potential issues are identified and addressed.
- Collaboration: A good integration test should involve collaboration among different teams and stakeholders, including developers, testers, and business analysts. This ensures that all relevant perspectives are considered and that the test is designed to effectively evaluate the integration of the system.
- Automation: A good integration test should be automated, to the extent possible, to save time and effort. This allows the test to be run repeatedly and consistently, without the need for manual intervention.
- Maintenance: A good integration test should be maintainable so that it can be easily updated and modified as the system evolves. This ensures that the test remains relevant and effective over time.

While a good unit test is one that effectively evaluates the functionality and behaviour of a single unit or component of a system. Several key characteristics make a good unit test:

- Isolation: A good unit test focuses on a single unit or component of the system and verifies its behaviour and functionality in isolation from the rest of the

system. This helps to ensure that the test is focused and relevant and that any issues or defects are specific to the unit being tested.

- Coverage: A good unit test should cover a wide range of scenarios and test cases, including both positive and negative test cases. This helps to ensure that the unit is thoroughly tested and that all potential issues are identified and addressed.
- Readability: A good unit test should be easy to read and understand, even for someone who is not familiar with the code, or the system being tested. This allows the test to be easily maintained and modified over time.
- Automation: A good unit test should be automated, to the extent possible, to save time and effort. This allows the test to be run repeatedly and consistently, without the need for manual intervention.
- Maintenance: A good unit test should be maintainable so that it can be easily updated and modified as the code or the system evolves. This ensures that the test remains relevant and effective over time.

2. **Given the following method specification, write down what tests (i.e. what test objectives each test would have and what initial conditions each would start with) you would need in order to test it comprehensively**

(wear, take) = decideWhatToWearAndTake(double temperature, double humidity)

if humidity is less than zero or greater than 100

throw an invalid parameter exception referring to invalid humidity

if the temperature is below 10 degrees and humidity is over 70%

wear a coat and take an umbrella

else if humidity is over 70%

take an umbrella

else if temperature is below 10 degrees

wear a wear a coat

else

wear a hat and take sunglasses

- Test for humidity < 0

Initial Conditions:

  - humidity = -5
  - temperature = 5

➢ **Test Objective:**

The objective of this test is to see if the exception is thrown with the above values passed into the function.

- Test for humidity > 100

Initial Conditions:

  - humidity = 105
  - temperature = 5

➢ **Test Objective:**

The objective of this test is to see if the exception is thrown with the above values passed into the function.

- Test for temperature < 10 && humidity > 70

Initial Conditions:

  - humidity = 80
  - temperature = 5

➢ **Test Objective:**

The objective for this test is to check the wear value = coat and the take value = umbrella at the end of the method.

- Test for temperature > 10 && humidity > 70

Initial Conditions:

  - humidity = 80
  - temperature = 15

➢ **Test Objective:**

The objective for this test is to test that the take value = umbrella.

- Test for temperature < 10 && humidity < 70

Initial Conditions:

  - humidity = 50
  - temperature = 5

➢ **Test Objective:**

The objective for this test is to test that the wear value = coat.

- Test for temperature > 10 && humidity < 70

Initial Conditions:

  - humidity = 50
  - temperature = 25

➢ **Test Objective:**

The objective for this test is to test that the wear value = hat and the take value = sunglasses.

3. What else have you learned in regard to dynamic testing?

I have learned that dynamic testing includes several different types of tests that all involve executing a program to identify errors and bugs. The testing is done to evaluate the behaviour of the software under different conditions. It is used to ensure that the software is functioning correctly and meets the requirements of the user. Dynamic testing is important because it allows testers to identify errors that may not be visible in other types of testing, such as static testing.

Dynamic testing and static testing are two different approaches to evaluating the quality of a software program. Dynamic testing involves executing the software and subjecting it to various inputs to evaluate its behaviour and outputs, while static testing involves analysing the code and the structure of the software without executing it.

I have also learned about White-Box and Black-Box testing techniques. Black-box testing involves testing the software without any knowledge of its internal workings, while white-box testing involves testing the software with knowledge of its internal structure and design.

Dynamic testing is often used in two main testing levels: integration and system testing. However, dynamic testing can also be used at other testing levels, such as unit testing and acceptance testing. Acceptance testing is performed by the end-user or customer to ensure that the system meets their needs and expectations.