|  |  |
| --- | --- |
| Course: | INFO-6067 Testing for Development |
| Project: | Project #1 – *Test Patterns* |

|  |  |
| --- | --- |
| Professor: | **Andy Wing** |

|  |  |
| --- | --- |
| Student Name: | *\_\_*Amanjot kaur*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
|  |  |
| Student ID#: | \_\_\_\_\_\_0914790\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Ans 1** : **Test Pattern**

The need to introduce the Test Pattern is to test the overall functionality of a class that has more than one methods. The main purpose of Testing pattern is to make the use of testing processes in a more reliable and optimum way.

In Java, Testing means creating Junit class for every class. The basic syntax of assertion pattern in Junit is to test the single method instead of whole class that might not be effective.

In this case, Test Pattern plays an important role to test multiple methods or whole class.

In Selection, Testing pattern should met customer’s requirements. Various criteria should taken for this purpose like risk, quality metrics, budget etc.

**Benefits of Test Pattern :**

1. **Save time**: Test pattern helps the developer and tester to reuse the same modules again and again for same repetitive task that saves their a lot of time.
2. **Fulfill different Requirement for single system :** Documentation prepared in test pattern like pattern contracts, pattern test case template and many other documentation satisfies different needs (compatibility, efficiency and effectiveness) for every system.
3. **Improves Quality**: Test pattern produces specific test cases for particular quality goals and combine them into one test suite. These test suites have different test cases that have different quality goals that directly or indirectly improves the performance and quality of requirements.
4. Test patterns provides adaptable test cases that can easily be updated and refactored in different versions. This whole process reduces maintenance costs.

**Drawbacks of** **Test Pattern :**

1.) New Test parameter in testing environment results in editing of existing requirements and test cases that might cause issues in their formats For e.g : JSON is more adaptable format than CSV to produce output.

2) Degrade overall performance: Most of the tests are repetitive in nature and if one of the tests has any error, it would probably not beneficial for the overall application.

**Example of Testing Pattern**

**Database Testing Pattern :**

**WHY USE DATABASE TESTING PATTERN ?**

Due to the growing data, it is difficult to manage all the data in single file. To manage such a voluminous data. New data loaded into the memory, then processed and printed on screen as output.

To handle such a large data, primary keys are retrieved at the beginning of execution and combination of those keys are used to consume information for various purposes. This process continued repeat itself.

Mocking the database : (First approach)

Use of Select() and Exec() to wrap up database library to make it as interfaces for database. Map are used as a SQL backup that help in unit testing as a alternative implementation.

Advantages :

1. It solve problems instantly and take input from user.
2. Coverage level can be satisfied by creating testcases for database functions.

Disadvantage :

1. It deal with code coverage that can’t be considered as matrix for test quality.
2. The degree of output correctness couldn’t validate.

Interfaces: (Second approach )

Under this approach, database calls are encapsulated, also combine methods and data members.

Each function in a interfaces has two phrases : one is for input and other is for output. There are two line included for function definition, one is for input and other is for output. The visibility of function in interfaces kept as local. Combine all the interfaces and associate function with interfaces.

Output is one or several smaller structure that handled all the production logic for testing.

Advantage :

1. Concentrate all the external calls at unique place.
2. This interface do not deal with how data look like or the structure from where data comes. It entirely wiped out the reflection concept.

**HSQLDB :**

1. DbUnitTestExecutionListener used to establish connection prior execution.
2. @DatabaseSetup annotation using fixture data define database.
3. Execute query against actual database.

**PART : 2 COMMON TESTING PATTERN**

**Observer Test pattern :**

Observer Test pattern is to test the observer pattern. To understand the concept of observer Test Pattern, First there is need to understand the concept of Observer pattern.

**The Observer Pattern** :

Four entities are there in observer pattern : Observer, Subject, concrete Subject, concrete Observer. Subject has multiple dependent observers(objects). Observer gets notification at the moment subject changes its state.

Subject is the central object whose state is monitored, due to which observer take some actions. Concrete Subject and Concrete Observer are actual subject and actual object on which abstract subject and abstract observer dependent.

**The Observer Test Pattern is created to test the above Observer Test Pattern. Its purpose is to** :

1)Is the data between subject and object are consistent or not ?

2)To check the validity of queries send by Observers .

3)Are notification send y Subject to observer are complete or not ?

**How it works :**

The implementation of subject and object need to be tested separately. Create Test class for each of the application class : Subject/observer class, and observer tester objects for concrete classes :

Subject class, Observer class, Concrete\_subject class, Concrete\_Observer class,

observerTest.

ObserverTest class refers both Concrete\_subject class, Concrete\_Observer class.

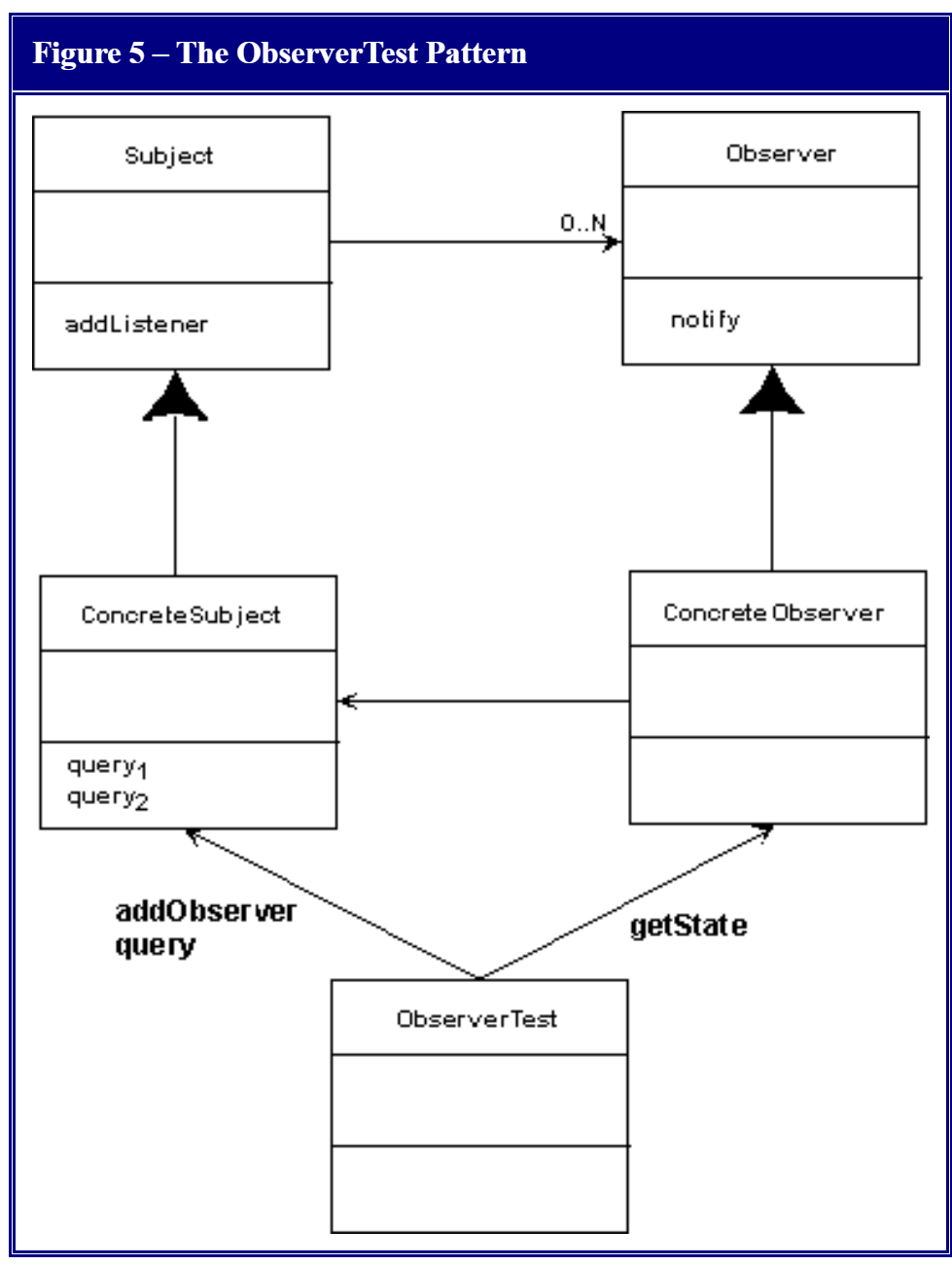
Create the object of Both the class to access its methods. With the created object of concrete\_suject class, add observer query. Similarly, Get the information of state from concrete\_observer class.

When observerTest object calls subject method, it change the state and provide information about notification. On basis of this, it will check two things

a)will it sent notification

b)will observer object perform its functions or not ?

To check the interaction between subject and object, various methods can be used that include interfaces, test cases, test suites.



Evaluation of Observer Test Pattern :

1. Observer test pattern provide the effective technique through which tests are created to test the interaction between the classes like interaction between subject and object.
2. It uses multiple observers one for every process.

**EXAMPLE FOR ACCEPTED CHANGES :**

1.)First observer create ‘Vetoable Change Listener’ in first step.

2.)Observer creates ‘Property change listener’ .

3)Observer add ‘Vetoable Change Listener’ to subject.

4)While observer add ‘Property change listener’ to subject, Subject respond by changing the state and send a message to observer through ‘Vetoable Change Listener’

5)In the last step, Subject make changes in properties or attributes and send a object-specific message to Observer through ‘Property change listener’.

**EXAMPLE FOR ACCEPTED CHANGES :**

1.)First observer create ‘Vetoable Change Listener’ in first step.

2.)Observer creates ‘Property change listener’ .

3)Observer add ‘Vetoable Change Listener’ to subject.

4)While observer add ‘Property change listener’ to subject, Subject respond by changing the state and send a message to observer through ‘Vetoable Change Listener’

5) A Exception is thrown by Observer called PropertyChangeException to Subject.

<https://sqa.stackexchange.com/questions/13280/data-driven-testing-patterns-pros-and-cons-and-good-examples>

<https://www.researchgate.net/publication/281772544_Testing_Patterns_in_Action_Designing_a_Test-Pattern-Based_Suite>

<https://people.cs.clemson.edu/~johnmc/joop/col18/column18.html>