LAB-2.

Test driven Development

Pre-Lab

1 Give an example of a situation where TDD will not work. (Hint: Think of situations where entire production code is required before starting the testing phase)

Soln:-

Our system takes in values from an external source. Typically there is a one to one mapping between an external value, and the corresponding internal value. However, now there are case where we need to take in two different values for a given internal value

Sometimes those external services which you have used contain most of the logic of a function. If your unit "grabs an item from a database using a query and returns it, using that interaction is 100% useless except as a substitute for type checking

2. Compare TDD with traditional testing

In TDD, the process decreases the time needed to launch and build the project, especially during cold starts

In TDD, test changes before changing the functionality which allows them to perform better

TDD approach project legacy while simplifying writing tests.

TDD makes the ~~cold~~ code clean

In Lab

① Write a Java Program for calculator by passing all test cases.

Step 1

* Calculator.java

```
public class Calculator
{
  public int add(int a, int b)
  { return a+b; }
public int sub(int a, int b)
  { return a-b; }
public int mul(int a, int b)
  { return a*b; }
public int div(int a, int b)
  { return a/b
```

```
try {
        return a/b;
    }

    catch (Exception e)
    {
    System.out.println("0 cannot be written as
                denominator");
    }

    return a/b;
    }
}
```

Step 2

* Test.java

```
public class Test {
    Calculator calculator = new Calculator();

    @Test
    public void testadd()
    { assertEquals(30, calculator.add(20, 10)); }

    @Test
    public void testsub()
    { assertEquals(40, calculator.sub(90, 50));
    }

    @Test
    public void testmul()
    { assertEquals(6, calculator.mul(2,3));

    }

    @Test
    public void testdiv()
    { assertEquals(25, calculator.div(50, 2));
    }
}
```

2. A Simple scenario: As a developer I want to implement code so that it prints the numbers from 1 to 100.

Given - an input of numbers from 1-100

when:

A number is a multible of '3' return "Fizz"

A number is a of '5' return 'Buzz';

A number is a of both '3' and '5' return "FizzBuzz"

A number is not divisible by '3' or '5' return the number itself

Then: print "Fizz", "Buzz", "FizzBuzz" or the number accordingly

Expected output - 1, 2, Fizz, 4, Buzz, ... 14, FizzBuzz, 16, ..

* FizzBuzz.java

```
public class FizzBuzz
{ public static void main (String [] args )
{ int n=100;
  for (int i=1; i<=n; i++)
  { if (i%.15 == 0)
    system.out.println ("FizzBuzz");
  else if(i %. 5==0)
  system.out.println ("Buzz");
  else if ( i%.3 ==0)
  system.out.println ("Fizz");
  else
  system.out.println (i);
  } } }
```

3. Write a Test driven program to accept the password when the length of it should be between 5 to 10 characters ("password validator")

Input
Abc12 3

Output
Valid password: accepted

* Password_Validator.java

```
import java.util.regex.Matcher;
public class Password_Validator
{
public static boolean isValidPassword(String Password)
  {
    String regex = "(?=\\S+$).{8,20}$";
    pattern p= Pattern.compile(regex);
      if(password ==null)
       { return false; }
    matcher m = p.matcher(password);
      return m.matches();
  }
}
```

* test.java

```
public class Test
  {
   @Test
   public void test()
     { password_Validator a = new Password_Validator();
   assertEquals(true, a.isValidPassword("Aman@123"));
   }
}
```