JUNIT ASSIGNMENT

Name:Shruti Bhosale

1)MinMaxFinder:

```java
import java.util.Arrays;

public class MinMaxFinder{
    public int[] findMinMax(int[] arr) {
        Arrays.sort(arr);
        int min=arr[0];
        int max=arr[arr.length-1];
        int[] result= {min,max};
        return result;

    }
}
```

Finished after 0.344 seconds

Runs: 3/3 Errors: 0 Failures: 0

- MinMaxFinderTest [Runner: JUnit 5] (0.001 s)
  - test3() (0.001 s)
  - testforNotEqual() (0.000 s)
  - testforEqual() (0.000 s)

Failure Trace

```java
import static org.junit.jupiter.api.Assertions.*;

class MinMaxFinderTest {
    MinMaxFinder minMaxFinder;
    @BeforeEach
    void init() {
        minMaxFinder=new MinMaxFinder();
    }
    int[] arr1= {50,80,60,30,20,10,40};
    @Test
    void testforEqual() {
       int[] expected= {10,80};
       int[] actual=minMaxFinder.findMinMax(arr1);
       assertArrayEquals(expected,actual);
    }

    @Test
    void testforNotEqual() {
       int[] expected= {10,50};
       int[] actual=minMaxFinder.findMinMax(arr1);
       Assertions.assertNotEquals(expected, actual);
    }

    @Test
    void test3() {
        boolean res=true;
        int[] expected=new int[] {10,80};
        int[] actual=minMaxFinder.findMinMax(arr1);
        if(actual!=null) {
            res=false;
        }
        Assertions.assertFalse(res);
    }

}
```

2)MinMaxFinder2:

```java
import java.util.Arrays;

public class MinMaxFinder2{

    public static int getMin(int[] arr) {
        Arrays.sort(arr);
        int min=arr[0];
        return min;
        }
    public static int getMax(int[] arr) {
        Arrays.sort(arr);
        int max=arr[arr.length-1];
        return max;
    }
}
```

Finished after 0.294 seconds

Runs: 2/2 Errors: 0 Failures: 0

MinMaxFinder2Test [Runner: JUnit 5] (0.080 s)
- testMax() (0.076 s)
- testMin() (0.001 s)

Failure Trace

```java
import static org.junit.jupiter.api.Assertions.*;

class MinMaxFinder2Test {
    MinMaxFinder minMaxFinder;
    @BeforeEach
    void init() {
        minMaxFinder=new MinMaxFinder();
    }
    int[] arr1= {50,80,60,30,20,10,40};
    int[] arr2= {12,45,27,93,19,39,20};
    int[] arr3= {79,65,11,28,14,29,36};

    @Test
    void testMin() {
        int val1=MinMaxFinder2.getMin(arr1);
        int val2=MinMaxFinder2.getMin(arr2);
        int val3=MinMaxFinder2.getMin(arr3);
        assert(val1==10);
        assert(val2==12);
        assert(val3==11);

    }
    @Test
    void testMax() {
        int val1=MinMaxFinder2.getMax(arr1);
        int val2=MinMaxFinder2.getMax(arr2);
        int val3=MinMaxFinder2.getMax(arr3);
        assert(val1==80);
        assert(val2==93);
        assert(val3==79);

    }
}
```

3)Bank Account:

```java

public class BankAccount {

    private int balance;
    public BankAccount() {
        balance=0;
    }
    public int getBalance() {
        return balance;
    }
    public void withdraw(int amount) throws InsufficientBalanceException
    {
        if(amount>balance)
            throw new InsufficientBalanceException("Insufficent Balance");
        else
            balance=balance-amount;
    }
    public void deposit(int amount)
    {
        balance=balance+amount;
    }
}

```

```java
import static org.junit.jupiter.api.Assertions.*;

public class BankTest {
    BankAccount customer;

    @BeforeClass
    public static void test1() {
        System.out.println("Test Data setup for all Test");
        Assume.assumeTrue(false);

    }
    @AfterClass
    public static void test2() {
        System.out.println("Test data cleaned");
    }
    @Before
    public void InitilizeCustomer() {
        System.out.println("@ Before Get Called");
        customer=new BankAccount();
    }

}
```

```java

public class InsufficientBalanceException extends Exception {

    public InsufficientBalanceException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public InsufficientBalanceException(String message, Throwable cause, boolean enableSuppression,
            boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
        // TODO Auto-generated constructor stub
    }

    public InsufficientBalanceException(String message, Throwable cause) {
        super(message, cause);
        // TODO Auto-generated constructor stub
    }

    public InsufficientBalanceException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

    public InsufficientBalanceException(Throwable cause) {
        super(cause);
        // TODO Auto-generated constructor stub
    }

}
```

Finished after 0.194 seconds

Runs: 2/2 Errors: 0 Failures: 0

- ExceptionTest [Runner: JUnit 5] (0.027 s)
  - ValidateWithdrawnExceptionTest (0.013 s)
  - ValidateWithdrawExceptionMessageTest (0.

```java
import static org.junit.jupiter.api.Assertions.*;


public class ExceptionTest {

    @Test(expected=InsufficientBalanceException.class)
    public void ValidateWithdrawnExceptionTest() throws InsufficientBalanceException
    {

        BankAccount customer = new BankAccount();
        customer.withdraw(1000);
    }
    @SuppressWarnings("deprecation")
    @Rule
    public ExpectedException thrown=ExpectedException.none();

    @Test
    public void ValidateWithdrawExceptionMessageTest() throws InsufficientBalanceException
    {
        thrown.expect(InsufficientBalanceException.class);
        thrown.expectMessage("Insufficent Balance");
        BankAccount customer = new BankAccount();
        customer.withdraw(1000);
    }
}
```

4)Life Cycle:

```java

public class Math {
    public int add (int a , int b) {
        return a+b;
    }

    public Object divide(int a, int b) {

        return a/b;
    }
}
```

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.Test;
class MathTest {
    static Math math;
      @BeforeAll
      static void beforeAllInit() {
          System.out.println("1.BeforeAll Executed");
      }
      @BeforeEach
      void init() {
          math = new Math();
          System.out.println("2.BeforeEach Executed");
      }
      @Test
      void testAdd() {

          int expected = 2;
          int actual = math.add(1,1);
          assertEquals(expected, actual, " Addition of two numbers");
      }
      @Test
       void testDivide() {
          assertThrows(ArithmeticException.class, ()->math.divide(1,0), "Divide by zero should throw");
      }
      @AfterEach
      void cleanup() {
          System.out.println("3.Test case -> successful");
      }
      @AfterAll
        static void afterAllfunc() {
          System.out.println("4.The application is terminated");
      }
      }
```

Output:

```
Console   Problems   Debug Shell
<terminated> MathTest (1) [JUnit] C:\Program Files\Java\jdk-
1.Before All Executed
2.BeforeEach executed
3.Test case -> successful
2.BeforeEach executed
3.Test case -> successful
4.The application is terminated
```

```
Finished after 0.366 seconds

Runs: 2/2    Errors: 0    Failures: 0

MathTest [Runner: JUnit 5] (0.003 s)
    testAdd() (0.000 s)
    testDivide() (0.003 s)
```