**Java Testing junit – Tanaya Jadhav**

**1)**

import java.util.Scanner;  
  
class MinMaxFinder{  
 public static int[] find()  
 {  
 int[] numbers = new int[9];  
 Scanner scan = new Scanner(System.*in*);  
 System.*out*.println("Input 9 numbers");  
 for(int i=0;i<9;i++)  
 {  
 numbers[i] = scan.nextInt();  
 }  
 int min = numbers[0];  
 int max = numbers[0];  
 for(int i=0;i<9;i++)  
 {  
 if(min>numbers[i])  
 {  
 min = numbers[i];  
 }  
 if(max<numbers[i])  
 {  
 max = numbers[i];  
 }  
 }  
 int[] arr = {min,max};  
 return arr;  
  
  
  
 }  
  
}

import org.junit.Test;  
  
import static org.junit.Assert.*assertEquals*;  
  
public class MinMaxtesting {  
 @Test  
 public void Test(){  
 MinMaxFinder minMaxFinder = new MinMaxFinder();  
 int[] expected = new int[]{3, 56};  
 int[] actual = MinMaxFinder.*find*();  
 *assertEquals*(expected, actual);  
  
 }  
  
}

import org.junit.Test;  
  
import static org.junit.Assert.*assertEquals*;  
import static org.junit.Assert.*assertNotEquals*;  
  
public class MinMaxtesting {  
 @Test  
 public void Test(){  
 MinMaxFinder minMaxFinder = new MinMaxFinder();  
 int[] expected = new int[]{4, 56};  
 int[] actual = MinMaxFinder.*find*();  
 *assertNotEquals*(expected, actual);  
  
 }

}

**Input 9 numbers**

**56**

**34**

**7**

**3**

**54**

**3**

**34**

**34**

**53**

**{3,56}**

**3)**

public class Bankmain {  
 String name;  
 //int acc\_id;  
 int balance;  
 int withdraw;  
  
  
 public Bankmain(String name, int balance, int withdraw) {  
 super();  
 this.name = name;  
 //this.acc\_id = acc\_id;  
 this.balance = balance;  
 this.withdraw = withdraw;  
 }  
  
 public int getBalance() throws InsufficientBalanceException {  
 if(balance>withdraw) {  
 balance=balance-withdraw;  
 return balance;  
 }  
 else {  
 throw new InsufficientBalanceException("Insufficient Balance"+balance);  
 }  
 }  
  
  
}

class InsufficientBalanceException extends Exception{  
  
 public InsufficientBalanceException(String message) {  
 super(message);  
 }  
}

import static org.junit.jupiter.api.Assertions.*assertThrows*;  
import static org.junit.Assert.*assertThrows*;  
  
import org.junit.Test;  
import org.junit.function.ThrowingRunnable;  
import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.function.Executable;  
  
class BankmainTest {  
  
 @Test  
 public void test() {  
 Bankmain bankmain=new Bankmain("Tanaya", 1000, 1001);  
 Assertions.*assertThrows*(InsufficientBalanceException.class, (Executable) bankmain);  
 try {  
 System.*out*.println("Balance:"+bankmain.getBalance());  
 } catch (InsufficientBalanceException e) {  
 e.printStackTrace();  
 }  
  
 Throwable exception = *assertThrows*(InsufficientBalanceException.class,  
 (ThrowingRunnable) ()->{bankmain.getBalance();});  
 }  
  
}

**4)**

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;  
  
class MathsTest {  
 static Maths *maths*;  
  
 @BeforeAll  
  
 static void beforeAllInit() {  
 System.*out*.println("1.Before All Executed");  
 }  
  
 @BeforeEach  
 void init() {  
 *maths* = new Maths();  
 System.*out*.println("2.BeforeEach executed");  
 }  
  
  
 @Test  
 void testAdd() {  
  
 int expected = 2;  
 int actual = *maths*.add(1,1);  
 *assertEquals*(expected, actual, " Addition of two numbers");  
 }  
  
  
 @Test  
 void testDivide() {  
 *assertThrows*(ArithmeticException.class, ()->*maths*.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");  
 }  
}

public class Maths {  
 public int add(int a, int b) {  
 return a + b;  
 }  
  
 public Object divide(int i, int j) {  
  
 return i / j;  
 }  
}

**1.Before All Executed**

**2.BeforeEach executed**

**3.Test case -> successful**

**2.BeforeEach executed**

**3.Test case -> successful**

**4.The application is terminated**

**Process finished with exit code 0**