**11.JUnit for Reversing a Word: Write a JUnit test to reverse a word and**

**use assert statements to validate the output.**

### **WordReverser.**java

import java.util.Scanner;

public class WordReverser {

public String reverse(String word) {

return new StringBuilder(word).reverse().toString();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

WordReverser wr = new WordReverser();

System.out.print("Enter a word: ");

String word = sc.nextLine();

System.out.println("Reversed word: " + wr.reverse(word));

}

}

### WordReverserTest.java

import static org.junit.Assert.\*;import org.junit.Test;

public class WordReverserTest {

@Test

public void testReverse() {

WordReverser wr = new WordReverser();

assertEquals("olleh", wr.reverse("hello"));

assertEquals("avaJ", wr.reverse("Java"));

assertEquals("a", wr.reverse("a"));

}

}

**12.JUnit for String Comparison: Develop a JUnit test to compare two**

**strings and use assert statements to validate the output. CO3 PO3**

### **StringComparator.java**

import java.util.Scanner;

public class StringComparator {

public boolean areEqual(String s1, String s2) {

return s1.equals(s2);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

StringComparator comp = new StringComparator();

System.out.print("Enter first string: ");

String s1 = sc.nextLine();

System.out.print("Enter second string: ");

String s2 = sc.nextLine();

if (comp.areEqual(s1, s2)) {

System.out.println("Strings are equal.");

} else {

System.out.println("Strings are not equal.");

}

}

}

### StringComparatorTest.java

import static org.junit.Assert.\*;import org.junit.Test;

public class StringComparatorTest {

@Test

public void testAreEqual() {

StringComparator comp = new StringComparator();

assertTrue(comp.areEqual("test", "test"));

assertFalse(comp.areEqual("hello", "world"));

assertTrue(comp.areEqual("", ""));

}

}

**13.JUnit for Voting System: Create a JUnit test for a voting system and**

**use assert statements to verify the functionality through white-box**

**testing.**

### **VotingSyst**em.java

import java.util.Scanner;

public class VotingSystem {

public boolean canVote(int age) {

return age >= 18;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

VotingSystem vs = new VotingSystem();

System.out.print("Enter your age: ");

int age = sc.nextInt();

if (vs.canVote(age)) {

System.out.println("Eligible to vote.");

} else {

System.out.println("Not eligible to vote.");

}

}

}

### VotingSystemTest.java

import static org.junit.Assert.\*;import org.junit.Test;

public class VotingSystemTest {

@Test

public void testCanVote() {

VotingSystem vs = new VotingSystem();

assertTrue(vs.canVote(18));

assertTrue(vs.canVote(25));

assertFalse(vs.canVote(17));

assertFalse(vs.canVote(10));

}

}

**14.Simple Interest Program with Senior Citizen Rate: Write a Java CO3 PO3**

**program to calculate simple interest. If the customer is a senior citizen,**

**the interest rate is 12%; otherwise, it is 10%. Verify the output using**

**white-box testing with JUnit.**

### **SimpleInterest.java**

import java.util.Scanner;

public class SimpleInterest {

public double calculateSI(double principal, double time, boolean isSenior) {

double rate = isSenior ? 12 : 10;

return (principal \* rate \* time) / 100;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

SimpleInterest si = new SimpleInterest();

System.out.print("Enter principal: ");

double principal = sc.nextDouble();

System.out.print("Enter time (in years): ");

double time = sc.nextDouble();

System.out.print("Are you a senior citizen? (true/false): ");

boolean isSenior = sc.nextBoolean();

double interest = si.calculateSI(principal, time, isSenior);

System.out.println("Simple Interest: " + interest);

}

}

### SimpleInterestTest.java

import static org.junit.Assert.\*;import org.junit.Test;

public class SimpleInterestTest {

@Test

public void testCalculateSI() {

SimpleInterest si = new SimpleInterest();

assertEquals(1200.0, si.calculateSI(10000, 1, true), 0.01);

assertEquals(1000.0, si.calculateSI(10000, 1, false), 0.01);

assertEquals(2400.0, si.calculateSI(20000, 1, true), 0.01);

}

}

**15.Palindrome Check Program: Develop a program to check whether a number is a palindrome and validate the output using white-box testing with JUnit.**

**Palindrome.java**

import java.util.Scanner;

public class Palindrome {

public boolean isPalindrome(int number) {

int original = number;

int reversed = 0;

while (number > 0) {

int digit = number % 10;

reversed = reversed \* 10 + digit;

number = number / 10;

}

return original == reversed;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Palindrome p = new Palindrome();

System.out.print("Enter a number: ");

int num = sc.nextInt();

if (p.isPalindrome(num)) {

System.out.println(num + " is a Palindrome.");

} else {

System.out.println(num + " is NOT a Palindrome.");

}

}

}

**PalindromeTest.java**

import static org.junit.Assert.\*;import org.junit.Test;

public class PalindromeTest {

Palindrome p = new Palindrome();

@Test

public void testPalindromeTrue() {

assertTrue(p.isPalindrome(121));

}

@Test

public void testPalindromeFalse() {

assertFalse(p.isPalindrome(123));

}

@Test

public void testSingleDigit() {

assertTrue(p.isPalindrome(7));

}

@Test

public void testZero() {

assertTrue(p.isPalindrome(0));

}

}

**TestRunner.java**

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.runClasses(PalindromeTest.class);

for (Failure failure : result.getFailures()) {

System.out.println("Test Failed: " + failure.toString());

}

System.out.println("All tests passed: " + result.wasSuccessful());

}

}

16.**Decimal to Binary and Octal Conversion: Write a program to convert a decimal number to its binary and octal equivalents. Validate the output using white-box testing with JUnit.**

## **DecimalConverter.java**

import java.util.Scanner;

public class DecimalConverter {

// Method to convert decimal to binary

public String toBinary(int decimal) {

if (decimal == 0) return "0";

return Integer.toBinaryString(decimal);

}

// Method to convert decimal to octal

public String toOctal(int decimal) {

if (decimal == 0) return "0";

return Integer.toOctalString(decimal);

}

// Main method for user input

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

DecimalConverter converter = new DecimalConverter();

System.out.print("Enter a decimal number: ");

int decimal = sc.nextInt();

System.out.println("Binary: " + converter.toBinary(decimal));

System.out.println("Octal : " + converter.toOctal(decimal));

}

}

**DecimalConverterTest.java**

import static org.junit.Assert.\*;

import org.junit.Test;

public class DecimalConverterTest {

DecimalConverter converter = new DecimalConverter();

// Case: decimal = 0

@Test

public void testZero() {

assertEquals("0", converter.toBinary(0));

assertEquals("0", converter.toOctal(0));

}

// Case: positive decimal number

@Test

public void testPositiveNumber() {

assertEquals("1010", converter.toBinary(10)); // 10 -> binary 1010

assertEquals("12", converter.toOctal(10)); // 10 -> octal 12

}

// Case: another positive number

@Test

public void testAnotherPositive() {

assertEquals("1111101", converter.toBinary(125)); // 125 -> binary

assertEquals("175", converter.toOctal(125)); // 125 -> octal

}

// Case: negative number (should match Java’s built-in 2’s complement)

@Test

public void testNegativeNumber() {

assertEquals(Integer.toBinaryString(-5), converter.toBinary(-5));

assertEquals(Integer.toOctalString(-5), converter.toOctal(-5));

}

}

**TestRunner.java**

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.runClasses(DecimalConverterTest.class);

for (Failure failure : result.getFailures()) {

System.out.println("Test Failed: " + failure.toString());

}

System.out.println("All tests passed: " + result.wasSuccessful());

}

}

**17.Days to Years, Weeks, and Days Conversion: Write a program to convert a given number of days into years, weeks, and days. Validate the output using white-box testing with JUnit.**

## **DaysConverter.java**

import java.util.Scanner;

public class DaysConverter {

// Method to convert days into years, weeks, and days

public int[] convert(int totalDays) {

int years = totalDays / 365;

int weeks = (totalDays % 365) / 7;

int days = (totalDays % 365) % 7;

return new int[]{years, weeks, days};

}

// Main method for user input

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

DaysConverter converter = new DaysConverter();

System.out.print("Enter number of days: ");

int totalDays = sc.nextInt();

int[] result = converter.convert(totalDays);

System.out.println(totalDays + " days = " +

result[0] + " year(s), " +

result[1] + " week(s), " +

result[2] + " day(s)");

## }

## DaysConverterTest.java

import static org.junit.Assert.\*;import org.junit.Test;

public class DaysConverterTest {

DaysConverter converter = new DaysConverter();

// Case: 0 days

@Test

public void testZeroDays() {

int[] result = converter.convert(0);

assertArrayEquals(new int[]{0, 0, 0}, result);

}

// Case: less than a week (only days)

@Test

public void testOnlyDays() {

int[] result = converter.convert(5);

assertArrayEquals(new int[]{0, 0, 5}, result);

}

// Case: exactly 1 week

@Test

public void testOneWeek() {

int[] result = converter.convert(7);

assertArrayEquals(new int[]{0, 1, 0}, result);

}

// Case: weeks + extra days

@Test

public void testWeeksAndDays() {

int[] result = converter.convert(10);

assertArrayEquals(new int[]{0, 1, 3}, result); // 1 week + 3 days

}

// Case: exactly 1 year

@Test

public void testOneYear() {

int[] result = converter.convert(365);

assertArrayEquals(new int[]{1, 0, 0}, result);

}

// Case: year + weeks + days

@Test

public void testYearWeeksDays() {

int[] result = converter.convert(400);

assertArrayEquals(new int[]{1, 5, 0}, result); // 1 year + 5 weeks

}

}

## TestRunner.java

import org.junit.runner.JUnitCore;import org.junit.runner.Result;import org.junit.runner.notification.Failure;

public class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.runClasses(DaysConverterTest.class);

for (Failure failure : result.getFailures()) {

System.out.println("Test Failed: " + failure.toString());

}

System.out.println("All tests passed: " + result.wasSuccessful());

}

}

**18.Factorial Calculation Program: Write a program to find the factorial of a number and verify the output using white-box testing with JUnit.**

**Factorial.java**

import java.util.Scanner;

public class Factorial {

// Method to calculate factorial

public long factorial(int n) {

if (n < 0) {

throw new IllegalArgumentException("Factorial not defined for negative numbers");

}

long result = 1;

for (int i = 1; i <= n; i++) {

result \*= i;

}

return result;

}

// Main method for user input

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Factorial f = new Factorial();

System.out.print("Enter a number: ");

int num = sc.nextInt();

try {

System.out.println("Factorial of " + num + " = " + f.factorial(num));

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

}

}

**FactorialTest.java**

import static org.junit.Assert.\*;

import org.junit.Test;

public class FactorialTest {

Factorial f = new Factorial();

// Case: factorial of 0 = 1

@Test

public void testZero() {

assertEquals(1, f.factorial(0));

}

// Case: factorial of 1 = 1

@Test

public void testOne() {

assertEquals(1, f.factorial(1));

}

// Case: factorial of 5 = 120

@Test

public void testFive() {

assertEquals(120, f.factorial(5));

}

// Case: negative number (should throw exception)

@Test(expected = IllegalArgumentException.class)

public void testNegativeNumber() {

f.factorial(-3);

}

}

**TestRunner.java**

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.runClasses(FactorialTest.class);

for (Failure failure : result.getFailures()) {

System.out.println("Test Failed: " + failure.toString());

}

System.out.println("All tests passed: " + result.wasSuccessful());

}

}

**19.Leap Year Check Program: Develop a program to check whether a given year is a leap year. Verify the output using white-box testing with JUnit.**

****LeapYearCheck.java****

import java.util.Scanner;

public class LeapYearCheck {

// Logic method

public boolean isLeapYear(int year) {

// Leap year if divisible by 4 and (not divisible by 100 unless divisible by 400)

return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

// Main method for user input

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

LeapYearCheck ly = new LeapYearCheck();

System.out.print("Enter a year: ");

int year = sc.nextInt();

if (ly.isLeapYear(year)) {

System.out.println(year + " is a Leap Year.");

} else {

System.out.println(year + " is NOT a Leap Year.");

}

sc.close();

}

}

**LeapYearCheckTest.java**

import static org.junit.Assert.\*;import org.junit.Test;

public class LeapYearCheckTest {

@Test

public void testLeapYears() {

LeapYearCheck ly = new LeapYearCheck();

assertTrue(ly.isLeapYear(2000)); // Divisible by 400

assertTrue(ly.isLeapYear(2024)); // Divisible by 4, not 100

}

@Test

public void testNonLeapYears() {

LeapYearCheck ly = new LeapYearCheck();

assertFalse(ly.isLeapYear(1900)); // Divisible by 100 but not 400

assertFalse(ly.isLeapYear(2023)); // Not divisible by 4

}

}

**20.Square and Cube Calculation: Write a program to calculate the square and cube of a decimal number. Validate the output using white-box testing with JUnit.**

****SquareCubeCalc.java****

import java.util.Scanner;

public class SquareCubeCalc {

// Logic methods

public double square(double num) {

return num \* num;

}

public double cube(double num) {

return num \* num \* num;

}

// Main method

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

SquareCubeCalc calc = new SquareCubeCalc();

System.out.print("Enter a number: ");

double num = sc.nextDouble();

System.out.println("Square of " + num + " = " + calc.square(num));

System.out.println("Cube of " + num + " = " + calc.cube(num));

sc.close();

}

}

**SquareCubeCalcTest.java**

import static org.junit.Assert.\*;import org.junit.Test;

public class SquareCubeCalcTest {

@Test

public void testSquare() {

SquareCubeCalc calc = new SquareCubeCalc();

assertEquals(25.0, calc.square(5), 0.001);

assertEquals(0.25, calc.square(0.5), 0.001);

}

@Test

public void testCube() {

SquareCubeCalc calc = new SquareCubeCalc();

assertEquals(27.0, calc.cube(3), 0.001);

assertEquals(-8.0, calc.cube(-2), 0.001);

}

}