JUNIT PROGRAMS :

**11. JUnit Test for Reversing a Word**

**ReverseWord.java:**

import java.util.Scanner;

public class ReverseWord {

// Method to reverse a word

public static String reverseWord(String word) {

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

}

// Main method to take input and print output

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String input = sc.nextLine();

String reversed = reverseWord(input);

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

}

}

**ReverseWordTest.java:**

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class ReverseWordTest {

// Method to reverse a word

public static String reverseWord(String word) {

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

}

// JUnit Test Case

@Test

public void testReverseWord() {

assertEquals("avaJ", reverseWord("Java"));

assertEquals("tset", reverseWord("test"));

assertEquals("cba", reverseWord("abc"));

assertEquals("", reverseWord("")); // empty string case

}

}

**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(ReverseWordTest.class);

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

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

}

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

}

}

**12.JUnit Test for String Comparison**

**StringCompare.java:**

import java.util.Scanner;

public class StringCompare {

// Method to compare two strings

public static boolean areEqual(String s1, String s2) {

return s1.equals(s2);

}

// Main method to take input

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String str1 = sc.nextLine();

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

String str2 = sc.nextLine();

if (areEqual(str1, str2)) {

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

} else {

System.out.println("Strings are Not Equal");

}

}

}

**StringCompareTest.java:**

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class StringCompareTest {

@Test

public void testStringEquality() {

assertEquals(true, StringCompare.areEqual("hello", "hello")); // same string

assertEquals(false, StringCompare.areEqual("Hello", "hello")); // case-sensitive

assertEquals(false, StringCompare.areEqual("java", "python")); // different

assertEquals(true, StringCompare.areEqual("", "")); // empty strings

}

}

**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(StringCompareTest.class);

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

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

}

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

}

}

**13.JUnit Test for Voting System (White-Box Testing)**

**VotingSystem:**

import java.util.Scanner;

public class VotingSystem {

public static String checkEligibility(int age) {

if (age < 0) {

return "Invalid Age";

} else if (age < 18) {

return "Not Eligible to Vote";

} else {

return "Eligible to Vote";

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int age = sc.nextInt();

String result = checkEligibility(age);

System.out.println(result);

sc.close();

}

}

**VotingSystemTest:**

import static org.junit.Assert.\*;

import org.junit.Test;

public class VotingSystemTest {

@Test

public void testInvalidAge() {

assertEquals("Invalid Age", VotingSystem.checkEligibility(-5));

}

@Test

public void testUnderAge() {

assertEquals("Not Eligible to Vote", VotingSystem.checkEligibility(15));

}

@Test

public void testEligibleAge() {

assertEquals("Eligible to Vote", VotingSystem.checkEligibility(20));

}

@Test

public void testBoundaryAge() {

assertEquals("Eligible to Vote", VotingSystem.checkEligibility(18));

}

}

**VotingSystemTestRunner:**

public class VotingSystemTestRunner {

public static void main(String[] args) {

testInvalidAge();

testUnderAge();

testEligibleAge();

testBoundaryAge();

}

public static void testInvalidAge() {

String result = VotingSystem.checkEligibility(-5);

if(result.equals("Invalid Age")) {

System.out.println("testInvalidAge PASSED");

} else {

System.out.println("testInvalidAge FAILED");

}

}

public static void testUnderAge() {

String result = VotingSystem.checkEligibility(15);

if(result.equals("Not Eligible to Vote")) {

System.out.println("testUnderAge PASSED");

} else {

System.out.println("testUnderAge FAILED");

}

}

public static void testEligibleAge() {

String result = VotingSystem.checkEligibility(20);

if(result.equals("Eligible to Vote")) {

System.out.println("testEligibleAge PASSED");

} else {

System.out.println("testEligibleAge FAILED");

}

}

public static void testBoundaryAge() {

String result = VotingSystem.checkEligibility(18);

if(result.equals("Eligible to Vote")) {

System.out.println("testBoundaryAge PASSED");

} else {

System.out.println("testBoundaryAge FAILED");

}

}

}

**14.Simple Interest Program with Senior Citizen Rate (White-Box Testing)**

**SimpleInterestCalculator:**

import java.util.Scanner;

public class SimpleInterestCalculator {

// Calculates simple interest with senior citizen benefit

public static String calculate(double principal, double rate, double time, boolean isSenior) {

if (principal < 0 || rate < 0 || time < 0) {

return "Invalid";

}

// Senior citizens get extra 2% interest

if (isSenior) {

rate += 2.0;

}

double si = (principal \* rate \* time) / 100;

return String.valueOf(si);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

double principal = sc.nextDouble();

System.out.print("Enter Rate of Interest: ");

double rate = sc.nextDouble();

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

double time = sc.nextDouble();

System.out.print("Is Senior Citizen? (yes/no): ");

String seniorInput = sc.next();

boolean isSenior = seniorInput.equalsIgnoreCase("yes");

System.out.println("Simple Interest = " + calculate(principal, rate, time, isSenior));

sc.close();

}

}

**SimpleInterestCalculatorTest:**

import static org.junit.Assert.\*;

import org.junit.Test;

public class SimpleInterestCalculatorTest {

@Test

public void testNormalCase() {

assertEquals("1000.0", SimpleInterestCalculator.calculate(5000, 5, 4, false));

// SI = (5000\*5\*4)/100 = 1000

}

@Test

public void testSeniorCitizenCase() {

assertEquals("1200.0", SimpleInterestCalculator.calculate(5000, 5, 4, true));

// rate = 5 + 2 = 7 => SI = (5000\*7\*4)/100 = 1400

}

@Test

public void testZeroPrincipal() {

assertEquals("0.0", SimpleInterestCalculator.calculate(0, 5, 5, false));

}

@Test

public void testInvalidValues() {

assertEquals("Invalid", SimpleInterestCalculator.calculate(-1000, 5, 2, false));

}

@Test

public void testZeroRate() {

assertEquals("0.0", SimpleInterestCalculator.calculate(5000, 0, 3, false));

}

}

**TestRunner:**

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(SimpleInterestCalculatorTest.class);

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

System.out.println(f.toString());

}

System.out.println("Tests run: " + result.getRunCount() +

", Failures: " + result.getFailureCount() +

", Ignored: " + result.getIgnoreCount());

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

}

}

**15. Palindrome Check Program (White-Box Testing)**

**Palindrome code:**

import java.util.Scanner;

public class PalindromeCheck {

// Method to check if a number is palindrome

public static boolean isPalindrome(int num) {

int original = num;

int reversed = 0;

while (num != 0) {

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

return original == reversed;

}

// Main method for online compiler

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

if (isPalindrome(n)) {

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

} else {

System.out.println(n + " is Not a Palindrome");

}

}

}

**PalindromeTest.java:**

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class PalindromeTest {

@Test

public void testPalindromeNumbers() {

assertEquals(true, PalindromeCheck.isPalindrome(121)); // normal palindrome

assertEquals(false, PalindromeCheck.isPalindrome(123)); // non-palindrome

assertEquals(true, PalindromeCheck.isPalindrome(1221)); // even-length palindrome

assertEquals(false, PalindromeCheck.isPalindrome(1234)); // edge: not palindrome

assertEquals(true, PalindromeCheck.isPalindrome(0)); // edge: single digit

}

}

**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 (White-Box Testing)**

**DecimalConverter:**

import java.util.Scanner;

public class DecimalConverter {

public static String toBinary(int n) {

if (n < 0) return "Invalid";

return Integer.toBinaryString(n);

}

public static String toOctal(int n) {

if (n < 0) return "Invalid";

return Integer.toOctalString(n);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int num = sc.nextInt();

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

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

sc.close();

}

}

**DecimalConverterTest:**

import static org.junit.Assert.\*;

import org.junit.Test;

public class DecimalConverterTest {

// --- Binary tests (cover n < 0 and n >= 0 paths) ---

@Test

public void testBinaryZero() {

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

}

@Test

public void testBinaryPositiveSmall() {

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

}

@Test

public void testBinaryEdgeByte() {

assertEquals("11111111", DecimalConverter.toBinary(255)); // 255 -> 8 ones

}

@Test

public void testBinaryNegative() {

assertEquals("Invalid", DecimalConverter.toBinary(-1));

}

// --- Octal tests (cover n < 0 and n >= 0 paths) ---

@Test

public void testOctalZero() {

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

}

@Test

public void testOctalPositiveSmall() {

assertEquals("14", DecimalConverter.toOctal(12)); // 12 -> 14 (octal)

}

@Test

public void testOctalEdgeByte() {

assertEquals("377", DecimalConverter.toOctal(255)); // 255 -> 377 (octal)

}

@Test

public void testOctalNegative() {

assertEquals("Invalid", DecimalConverter.toOctal(-5));

}

}

**TestRunner:**

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 f : result.getFailures()) {

System.out.println(f.toString());

}

System.out.println("Tests run: " + result.getRunCount() +

", Failures: " + result.getFailureCount() +

", Ignored: " + result.getIgnoreCount());

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

}

}

**17.Days to Years, Weeks, and Days Conversion (White-Box Testing)**

**DaysConverter:**

import java.util.Scanner;

public class DaysConverter {

// Converts days into years, weeks, and days

public static String convert(int days) {

if (days < 0) return "Invalid";

int years = days / 365;

int weeks = (days % 365) / 7;

int remDays = (days % 365) % 7;

return years + " Years, " + weeks + " Weeks, " + remDays + " Days";

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int days = sc.nextInt();

System.out.println("Converted: " + convert(days));

sc.close();

}

}

**DaysConverterTest:**

import static org.junit.Assert.\*;

import org.junit.Test;

public class DaysConverterTest {

@Test

public void testZeroDays() {

assertEquals("0 Years, 0 Weeks, 0 Days", DaysConverter.convert(0));

}

@Test

public void testNormalDays() {

assertEquals("1 Years, 1 Weeks, 1 Days", DaysConverter.convert(373));

// 373 = 365 + 7 + 1

}

@Test

public void testOnlyWeeks() {

assertEquals("0 Years, 2 Weeks, 0 Days", DaysConverter.convert(14));

}

@Test

public void testOnlyDays() {

assertEquals("0 Years, 0 Weeks, 3 Days", DaysConverter.convert(3));

}

@Test

public void testNegativeDays() {

assertEquals("Invalid", DaysConverter.convert(-10));

}

}

**TestRunner:**

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 f : result.getFailures()) {

System.out.println(f.toString());

}

System.out.println("Tests run: " + result.getRunCount() +

", Failures: " + result.getFailureCount() +

", Ignored: " + result.getIgnoreCount());

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

}

}

**18.Factorial Calculation Program (White-Box Testing)**

**FactorialCalculator:**

import java.util.Scanner;

public class FactorialCalculator {

// Returns factorial of n, handles invalid inputs

public static String factorial(int n) {

if (n < 0) return "Invalid"; // negative case

if (n == 0 || n == 1) return "1"; // base cases

int fact = 1;

for (int i = 2; i <= n; i++) {

fact \*= i;

}

return String.valueOf(fact);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int num = sc.nextInt();

System.out.println("Factorial: " + factorial(num));

sc.close();

}

}

**FactorialCalculatorTest:**

import static org.junit.Assert.\*;

import org.junit.Test;

public class FactorialCalculatorTest {

@Test

public void testFactorialZero() {

assertEquals("1", FactorialCalculator.factorial(0)); // 0! = 1

}

@Test

public void testFactorialOne() {

assertEquals("1", FactorialCalculator.factorial(1)); // 1! = 1

}

@Test

public void testFactorialSmall() {

assertEquals("120", FactorialCalculator.factorial(5)); // 5! = 120

}

@Test

public void testFactorialLarge() {

assertEquals("3628800", FactorialCalculator.factorial(10)); // 10! = 3628800

}

@Test

public void testNegativeInput() {

assertEquals("Invalid", FactorialCalculator.factorial(-4)); // invalid

}

}

**TestRunner:**

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(FactorialCalculatorTest.class);

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

System.out.println(f.toString());

}

System.out.println("Tests run: " + result.getRunCount() +

", Failures: " + result.getFailureCount() +

", Ignored: " + result.getIgnoreCount());

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

}

}

**19.Leap Year Check Program (White-Box Testing)**

**LeapYearChecker:**

import java.util.Scanner;

public class LeapYearChecker {

// Returns whether a year is leap year or not

public static String checkLeapYear(int year) {

if (year < 0) return "Invalid";

if ((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0)) {

return "Leap Year";

} else {

return "Not a Leap Year";

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int year = sc.nextInt();

System.out.println("Result: " + checkLeapYear(year));

sc.close();

}

}

**LeapYearCheckerTest:**

import static org.junit.Assert.\*;

import org.junit.Test;

public class LeapYearCheckerTest {

@Test

public void testLeapYearDivBy400() {

assertEquals("Leap Year", LeapYearChecker.checkLeapYear(2000));

}

@Test

public void testLeapYearDivBy4Not100() {

assertEquals("Leap Year", LeapYearChecker.checkLeapYear(2024));

}

@Test

public void testNotLeapYearDivBy100Not400() {

assertEquals("Not a Leap Year", LeapYearChecker.checkLeapYear(1900));

}

@Test

public void testNotLeapYearOther() {

assertEquals("Not a Leap Year", LeapYearChecker.checkLeapYear(2023));

}

@Test

public void testInvalidYear() {

assertEquals("Invalid", LeapYearChecker.checkLeapYear(-500));

}

}

**TestRunner:**

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(LeapYearCheckerTest.class);

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

System.out.println(f.toString());

}

System.out.println("Tests run: " + result.getRunCount() +

", Failures: " + result.getFailureCount() +

", Ignored: " + result.getIgnoreCount());

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

}

}

**20.Square and Cube Calculation Program (White-Box Testing)**

**PowerCalculator:**

import java.util.Scanner;

public class PowerCalculator {

// Returns square of n

public static String square(int n) {

return String.valueOf(n \* n);

}

// Returns cube of n

public static String cube(int n) {

return String.valueOf(n \* n \* n);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int num = sc.nextInt();

System.out.println("Square: " + square(num));

System.out.println("Cube : " + cube(num));

sc.close();

}

}

**PowerCalculatorTest:**

import static org.junit.Assert.\*;

import org.junit.Test;

public class PowerCalculatorTest {

@Test

public void testSquarePositive() {

assertEquals("25", PowerCalculator.square(5)); // 5^2 = 25

}

@Test

public void testSquareZero() {

assertEquals("0", PowerCalculator.square(0)); // 0^2 = 0

}

@Test

public void testSquareNegative() {

assertEquals("9", PowerCalculator.square(-3)); // (-3)^2 = 9

}

@Test

public void testCubePositive() {

assertEquals("27", PowerCalculator.cube(3)); // 3^3 = 27

}

@Test

public void testCubeZero() {

assertEquals("0", PowerCalculator.cube(0)); // 0^3 = 0

}

@Test

public void testCubeNegative() {

assertEquals("-8", PowerCalculator.cube(-2)); // (-2)^3 = -8

}

}

**TestRunner:**

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(PowerCalculatorTest.class);

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

System.out.println(f.toString());

}

System.out.println("Tests run: " + result.getRunCount() +

", Failures: " + result.getFailureCount() +

", Ignored: " + result.getIgnoreCount());

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

}

}