Concepts of OOP

1. Write a program to demonstrate encapsulation in Java.

class Person {

private String name;

private int age;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

if(age > 0) this.age = age;

}

}

1. Create a program showing the use of inheritance and polymorphism.

class Animal {

void sound() {

System.out.println("Animal makes sound");

}

}

class Dog extends Animal {

void sound() {

System.out.println("Dog barks");

}

}

class Cat extends Animal {

void sound() {

System.out.println("Cat meows");

}

}

1. Explain and implement the concept of abstraction in Java using interfaces.

interface Vehicle {

void start();

}

class Car implements Vehicle {

public void start() {

System.out.println("Car starts");

}

}

1. Write a program to demonstrate method overloading and method overriding.

class Calculator {

int add(int a, int b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

}

class AdvancedCalculator extends Calculator {

@Override

int add(int a, int b) {

return super.add(a, b) + 1; // Just for demonstration

}

}

1. Create a class hierarchy for animals that demonstrates polymorphism.

class Animal2 {

void makeSound() {

System.out.println("Some animal sound");

}

}

class Dog2 extends Animal2 {

void makeSound() {

System.out.println("Bark");

}

}

class Cat2 extends Animal2 {

void makeSound() {

System.out.println("Meow");

}

}

1. Develop a program to implement multiple inheritance using interfaces.

interface A {

void methodA();

}

interface B {

void methodB();

}

class C implements A, B {

public void methodA() {

System.out.println("Method A");

}

public void methodB() {

System.out.println("Method B");

}

}

1. Write a Java program to showcase the use of this and super keywords.

class Parent {

int x = 10;

}

class Child extends Parent {

int x = 20;

void show() {

System.out.println("x: " + this.x);

System.out.println("super.x: " + super.x);

}

}

1. Demonstrate the concept of constructors in OOP with a program.

class Student {

String name;

Student() {

name = "Unknown";

}

Student(String n) {

name = n;

}

}

1. Explain and implement the concept of access modifiers in Java

class AccessExample {

public int pub = 1;

protected int pro = 2;

int def = 3;

private int pri = 4;

void printAccess() {

System.out.println(pub + " " + pro + " " + def + " " + pri);

}

}

1. Show an example of the final keyword for variables, methods, and classes.

final class FinalClass {

final int value = 10;

final void display() {

System.out.println("Final method and class");

}

}

1. Write a program that uses Java's StringBuilder for efficient string operations.

class StringBuilderExample {

public static void main(String[] args) {

StringBuilder sb = new StringBuilder("Hello");

sb.append(" World");

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

}

}

1. Write a program to demonstrate the immutability of the String class.

class StringImmutable {

public static void main(String[] args) {

String s = "hello";

s.concat(" world");

System.out.println(s); // prints "hello"

}

}

Data Types, Control Statements, and Identifiers

1. Write a program to declare variables of all primitive data types in Java and print their default values.

public class PrimitiveDefaults {

static byte b;

static short s;

static int i;

static long l;

static float f;

static double d;

static char c;

static boolean bool;

public static void main(String[] args) {

System.out.println("byte: " + b);

System.out.println("short: " + s);

System.out.println("int: " + i);

System.out.println("long: " + l);

System.out.println("float: " + f);

System.out.println("double: " + d);

System.out.println("char: '" + c + "'");

System.out.println("boolean: " + bool);

}

}

1. Implement a program to demonstrate the use of if-else, switch, and for loops.

public class ControlStatements {

public static void main(String[] args) {

int num = 5;

// if-else

if (num % 2 == 0)

System.out.println("Even");

else

System.out.println("Odd");

// switch

switch (num) {

case 1: System.out.println("One"); break;

case 5: System.out.println("Five"); break;

default: System.out.println("Other");

}

// for loop

for (int i = 1; i <= 3; i++) {

System.out.println("Loop count: " + i);

}

}

}

1. Write a program to check if a number is prime using a while loop.

public class PrimeCheck {

public static void main(String[] args) {

int num = 29, i = 2;

boolean isPrime = true;

while (i <= num / 2) {

if (num % i == 0) {

isPrime = false;

break;

}

i++;

}

System.out.println(num + (isPrime ? " is Prime." : " is not Prime."));

}

}

1. Create a program to calculate the factorial of a number using recursion.

public class FactorialRecursion {

public static int factorial(int n) {

return (n == 0) ? 1 : n \* factorial(n - 1);

}

public static void main(String[] args) {

System.out.println("Factorial: " + factorial(5));

}

}

1. Write a program to identify valid and invalid identifiers in Java.

public class IdentifierTest {

int validName = 1;

int $alsoValid = 2;

int \_underscore = 3;

// int 123invalid = 4; // invalid: starts with digit

// int class = 5; // invalid: reserved keyword

// int @notValid = 6; // invalid: contains symbol

}

Arrays

1. Write a program to find the largest and smallest numbers in an array.

public class MinMaxArray {

public static void main(String[] args) {

int[] arr = {4, 2, 9, 1, 7};

int min = arr[0], max = arr[0];

for (int num : arr) {

if (num < min) min = num;

if (num > max) max = num;

}

System.out.println("Min: " + min + ", Max: " + max);

}

}

1. Write a program to check if a given number is odd or even.

public class EvenOdd {

public static void main(String[] args) {

int num = 10;

System.out.println(num + (num % 2 == 0 ? " is even." : " is odd."));

}

}

1. Write a program to find the largest of three numbers entered by the user.

import java.util.Scanner;

public class LargestOfThree {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter 3 numbers: ");

int a = sc.nextInt(), b = sc.nextInt(), c = sc.nextInt();

int max = (a > b) ? (a > c ? a : c) : (b > c ? b : c);

System.out.println("Largest: " + max);

}

}

1. Write a program to calculate the factorial of a given number using recursion.

public class Factorial {

public static int factorial(int n) {

return (n == 0) ? 1 : n \* factorial(n - 1);

}

public static void main(String[] args) {

System.out.println("Factorial: " + factorial(6));

}

}

1. Write a program to check if a given string or number is a palindrome.

public class Palindrome {

public static void main(String[] args) {

String str = "madam";

String rev = new StringBuilder(str).reverse().toString();

System.out.println(str + (str.equals(rev) ? " is a palindrome." : " is not a palindrome."));

}

}

1. Write a program to generate the first n terms of the Fibonacci series.

public class FibonacciSeries {

public static void main(String[] args) {

int n = 10, a = 0, b = 1;

System.out.print("Fibonacci: " + a + " " + b + " ");

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

int c = a + b;

System.out.print(c + " ");

a = b;

b = c;

}

}

}

1. Write a program to check whether a given number is prime.

public class ReverseString {

public static void main(String[] args) {

String str = "OpenAI";

String reversed = new StringBuilder(str).reverse().toString();

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

}

}

1. Write a program to find the sum of all elements in an array.

public class VowelConsonantCount {

public static void main(String[] args) {

String str = "Hello World";

int vowels = 0, consonants = 0;

str = str.toLowerCase();

for (char c : str.toCharArray()) {

if (Character.isLetter(c)) {

if ("aeiou".indexOf(c) != -1)

vowels++;

else

consonants++;

}

}

System.out.println("Vowels: " + vowels + ", Consonants: " + consonants);

}

}

1. Implement a program to reverse the elements of an array.

public class StringCaseConversion {

public static void main(String[] args) {

String str = "Java Programming";

System.out.println("Uppercase: " + str.toUpperCase());

System.out.println("Lowercase: " + str.toLowerCase());

}

}

1. Write a Java program to perform matrix addition and multiplication.

public class VowelConsonantCheck {

public static void main(String[] args) {

char c = 'e';

if (Character.isLetter(c)) {

if ("aeiouAEIOU".indexOf(c) != -1)

System.out.println(c + " is a vowel.");

else

System.out.println(c + " is a consonant.");

} else {

System.out.println("Not a letter.");

}

}

}

1. Create a program to sort an array using the bubble sort algorithm.

public class AsciiValue {

public static void main(String[] args) {

char ch = 'A';

int ascii = (int) ch;

System.out.println("ASCII value of " + ch + " is: " + ascii);

}

}

1. Write a program to demonstrate a 2D array and print its elements.

public class NumberCheck {

public static void main(String[] args) {

int num = -10;

if (num > 0)

System.out.println("Positive");

else if (num < 0)

System.out.println("Negative");

else

System.out.println("Zero");

}

}

1. Write a program to search for an element in a sorted array using the binary search algorithm.

public class MultiplicationTable {

public static void main(String[] args) {

int num = 5;

for (int i = 1; i <= 10; i++) {

System.out.println(num + " x " + i + " = " + (num \* i));

}

}

}

1. Write a program to remove duplicate elements from an array.

public class SwapNumbers {

public static void main(String[] args) {

int a = 10, b = 20;

System.out.println("Before Swap: a=" + a + " b=" + b);

a = a + b;

b = a - b;

a = a - b;

System.out.println("After Swap: a=" + a + " b=" + b);

}

}

Operators

1. Write a program to demonstrate the use of arithmetic, relational, and logical operators.

public class LeapYearCheck {

public static void main(String[] args) {

int year = 2024;

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

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

else

System.out.println(year + " is not a leap year.");

}

}

1. Create a program to show the difference between == and equals() for string comparison.

public class PowerCalculator {

public static void main(String[] args) {

int base = 2, exponent = 3;

int result = 1;

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

result \*= base;

}

System.out.println(base + " raised to " + exponent + " is: " + result);

}

}

1. Write a program to illustrate the use of the ternary operator.

public class DigitSum {

public static void main(String[] args) {

int num = 1234;

int sum = 0;

while (num != 0) {

sum += num % 10;

num /= 10;

}

System.out.println("Sum of digits: " + sum);

}

}

1. Implement a program to perform bitwise operations in Java.

public class CountDigits {

public static void main(String[] args) {

int num = 12345;

int count = 0;

while (num != 0) {

num /= 10;

count++;

}

System.out.println("Number of digits: " + count);

}

}

1. Write a program to demonstrate operator precedence in Java.

public class FactorialRecursive {

public static int factorial(int n) {

if (n == 0)

return 1;

else

return n \* factorial(n - 1);

}

public static void main(String[] args) {

int num = 5;

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

}

}

Classes and Constructors

1. Write a program to create a class with multiple constructors (constructor overloading).

public class PowerRecursive {

public static int power(int base, int exp) {

if (exp == 0)

return 1;

return base \* power(base, exp - 1);

}

public static void main(String[] args) {

System.out.println("2^5 = " + power(2, 5));

}

}

1. Implement a program to demonstrate the use of a copy constructor in Java.

public class GCD {

public static int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

public static void main(String[] args) {

System.out.println("GCD of 20 and 8 is: " + gcd(20, 8));

}

}

1. Create a program that initializes class fields using a parameterized constructor.

public class LCM {

public static int gcd(int a, int b) {

while (b != 0) {

int t = b;

b = a % b;

a = t;

}

return a;

}

public static int lcm(int a, int b) {

return (a \* b) / gcd(a, b);

}

public static void main(String[] args) {

System.out.println("LCM of 12 and 15 is: " + lcm(12, 15));

}

}

1. Write a program to demonstrate the use of static and non-static methods.

public class FibonacciRecursive {

public static int fibonacci(int n) {

if (n <= 1)

return n;

return fibonacci(n - 1) + fibonacci(n - 2);

}

public static void main(String[] args) {

int n = 10;

for (int i = 0; i < n; i++)

System.out.print(fibonacci(i) + " ");

}

}

1. Implement a singleton class in Java.

public class FibonacciLoop {

public static void main(String[] args) {

int n = 10, a = 0, b = 1;

System.out.print(a + " " + b + " ");

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

int next = a + b;

System.out.print(next + " ");

a = b;

b = next;

}

}

}

Inheritance and Method Overriding

1. Write a program to demonstrate multilevel inheritance in Java.

public class PrimeCheck {

public static void main(String[] args) {

int num = 29;

boolean isPrime = true;

for (int i = 2; i <= num / 2; i++) {

if (num % i == 0) {

isPrime = false;

break;

}

}

if (num <= 1) isPrime = false;

System.out.println(num + " is prime? " + isPrime);

}

}

1. Create a program to show method overriding and the use of super to call the parent class method.

public class PalindromeCheck {

public static void main(String[] args) {

int num = 121, reversed = 0, original = num;

while (num != 0) {

int digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

System.out.println(original + " is palindrome? " + (original == reversed));

}

}

1. Implement an abstract class and override its methods in a subclass.

public class ArmstrongCheck {

public static void main(String[] args) {

int num = 153, temp = num, sum = 0;

while (temp != 0) {

int digit = temp % 10;

sum += digit \* digit \* digit;

temp /= 10;

}

System.out.println(num + " is Armstrong? " + (sum == num));

}

}

1. Write a program to demonstrate final classes and methods.

public class PrimeNumbers1to100 {

public static void main(String[] args) {

for (int i = 2; i <= 100; i++) {

boolean isPrime = true;

for (int j = 2; j <= i / 2; j++) {

if (i % j == 0) {

isPrime = false;

break;

}

}

if (isPrime)

System.out.print(i + " ");

}

}

}

1. Create a program to show run-time polymorphism using dynamic method dispatch.

public class Factors {

public static void main(String[] args) {

int num = 28;

System.out.print("Factors of " + num + ": ");

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

if (num % i == 0)

System.out.print(i + " ");

}

}

}

String Class and Operations

1. Write a program to reverse a string without using built-in methods.
2. Implement a program to count the frequency of characters in a string.

public class AsciiAZ {

public static void main(String[] args) {

for (char c = 'A'; c <= 'Z'; c++) {

System.out.println(c + ": " + (int) c);

}

}

}

1. Write a program to demonstrate the immutability of the String class.

public class CharIntConversion {

public static void main(String[] args) {

char ch = 'A';

int ascii = (int) ch;

char ch2 = (char) ascii;

System.out.println("Char to int: " + ascii);

System.out.println("Int to char: " + ch2);

}

}

1. Create a program to check if a given string is a palindrome.

import java.util.HashMap;

public class CharFrequency {

public static void main(String[] args) {

String str = "hello world";

HashMap<Character, Integer> freq = new HashMap<>();

for (char c : str.toCharArray()) {

if (Character.isLetter(c)) {

freq.put(c, freq.getOrDefault(c, 0) + 1);

}

}

for (char c : freq.keySet()) {

System.out.println(c + ": " + freq.get(c));

}

}

}

1. Implement a program to split a string into words and print each word on a new line.

public class RemoveSpaces {

public static void main(String[] args) {

String str = "Java Programming is fun";

str = str.replaceAll("\\s", "");

System.out.println("After removing spaces: " + str);

}

}