

**Malla Reddy College of Engineering & Technology**

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with ‘A’ Grade)

**Java Programming Guide**

**Java Coding Questions**

**1. // Write a simple Java program to print "Hello World"**

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello World");

}

}

**2. // Demonstrate type casting in Java**

public class TypeCasting {

public static void main(String[] args) {

int i = 100;

double d = i; // implicit casting

int j = (int) d; // explicit casting

System.out.println("Double: " + d + ", Int: " + j);

}

}

**3. // Program to find the factorial using recursion**

public class Factorial {

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));

}

}

**4. // Create a class and demonstrate constructor overloading**

class Car {

String model;

Car() {

model = "Default";

}

Car(String m) {

model = m;

}

void show() {

System.out.println("Model: " + model);

}

}

**5. // Demonstrate inheritance in Java**

class Animal {

void sound() {

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

}

}

class Dog extends Animal {

void sound() {

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

}

}

**6. // Demonstrate abstract class and method**

abstract class Shape {

abstract void draw();

}

class Circle extends Shape {

void draw() {

System.out.println("Drawing Circle");

}

}

**7. // Interface implementation**

interface Drawable {

void draw();

}

class Rectangle implements Drawable {

public void draw() {

System.out.println("Drawing Rectangle");

}

}

**8. // Demonstrate method overloading**

class Overload {

int sum(int a, int b) { return a + b; }

double sum(double a, double b) { return a + b; }

}

**9. // Demonstrate method overriding**

class Parent {

void show() {

System.out.println("Parent");

}

}

class Child extends Parent {

void show() {

System.out.println("Child");

}

}

**10. // Using 'this' keyword**

class Student {

String name;

Student(String name) {

this.name = name;

}

void display() {

System.out.println("Name: " + name);

}

}

**11. // Creating and accessing a package**

// package mypack; (create a file in directory mypack)

public class MyClass {

public void msg() {

System.out.println("Hello from package");

}

}

**12. // Try-catch-finally block**

public class TryCatchDemo {

public static void main(String[] args) {

try {

int a = 10 / 0;

} catch (ArithmeticException e) {

System.out.println("Arithmetic Exception caught");

} finally {

System.out.println("Finally block executed");

}

}

}

**13. // Custom exception**

class MyException extends Exception {

MyException(String msg) {

super(msg);

}

}

14. // Creating thread using Runnable

class MyRunnable implements Runnable {

public void run() {

System.out.println("Thread running");

}

}

**15. // Synchronizing a method**

class Counter {

synchronized void increment() {

System.out.println("Incrementing");

}

}

**16. // Producer-consumer using wait/notify (simple demo)**

class Shared {

int data;

boolean flag = false;

synchronized void produce(int d) throws InterruptedException {

while(flag) wait();

data = d; flag = true; notify();

}

synchronized int consume() throws InterruptedException {

while(!flag) wait();

flag = false; notify(); return data;

}

}

**17. // ArrayList example**

import java.util.\*;

class ListExample {

public static void main(String[] args) {

ArrayList<String> list = new ArrayList<>();

list.add("Java"); list.add("Python");

for(String lang : list) System.out.println(lang);

}

}

**18. // File handling**

import java.io.\*;

class FileDemo {

public static void main(String[] args) throws IOException {

FileWriter fw = new FileWriter("test.txt");

fw.write("Hello Java"); fw.close();

}

}

**19. // JDBC Connection (MySQL example)**

import java.sql.\*;

class DBConnect {

public static void main(String[] args) throws Exception {

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db", "user", "pass");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("select \* from student");

while(rs.next()) System.out.println(rs.getString(1));

con.close();

}

}

**20. // Swing JFrame Example**

import javax.swing.\*;

class MyFrame {

public static void main(String[] args) {

JFrame f = new JFrame("Example");

JLabel l = new JLabel("Hello");

f.add(l); f.setSize(200, 100);

f.setVisible(true);

}

**21. // Program to check whether a string is a palindrome**

public class PalindromeCheck {

public static void main(String[] args) {

String str = "madam";

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

System.out.println(str.equals(rev.toString()) ? "Palindrome" : "Not Palindrome");

}

}

**22. // Reverse a number**

public class ReverseNumber {

public static void main(String[] args) {

int num = 1234, rev = 0;

while(num != 0) {

rev = rev \* 10 + num % 10;

num /= 10;

}

System.out.println("Reversed: " + rev);

}

}

**23. // Program to check if a number is prime**

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;

}

}

System.out.println(isPrime ? "Prime" : "Not Prime");

}

}

**24. // Find the Fibonacci series**

public class Fibonacci {

public static void main(String[] args) {

int a = 0, b = 1, c;

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

for (int i = 2; i < 10; i++) {

c = a + b;

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

a = b; b = c;

}

}

}

**25. // Count number of vowels and consonants in a string**

public class CountVowels {

public static void main(String[] args) {

String str = "Hello World".toLowerCase();

int vowels = 0, consonants = 0;

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

if (ch >= 'a' && ch <= 'z') {

if ("aeiou".indexOf(ch) != -1) vowels++;

else consonants++;

}

}

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

}

}

**26. // Convert binary to decimal**

public class BinaryToDecimal {

public static void main(String[] args) {

String binary = "1010";

int decimal = Integer.parseInt(binary, 2);

System.out.println("Decimal: " + decimal);

}

}

**27. // Find GCD of two numbers**

public class GCD {

public static void main(String[] args) {

int a = 20, b = 8;

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

System.out.println("GCD: " + a);

}

}

**28. // Count digits in a number**

public class CountDigits {

public static void main(String[] args) {

int num = 12345, count = 0;

while (num != 0) {

count++;

num /= 10;

}

System.out.println("Digits: " + count);

}

}

**29. // Calculate power of a number**

public class PowerCalculation {

public static void main(String[] args) {

int base = 2, exponent = 5, result = 1;

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

result \*= base;

}

System.out.println("Power: " + result);

}

}

**30. // Convert decimal to binary**

public class DecimalToBinary {

public static void main(String[] args) {

int decimal = 10;

String binary = Integer.toBinaryString(decimal);

System.out.println("Binary: " + binary);

}

}

**32. // Check Armstrong number**

public class ArmstrongNumber {

public static void main(String[] args) {

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

while (temp != 0) {

int digit = temp % 10;

sum += Math.pow(digit, 3);

temp /= 10;

}

System.out.println((sum == num) ? "Armstrong" : "Not Armstrong");

}

}

**33. // Find sum of digits of a number**

public class SumOfDigits {

public static void main(String[] args) {

int num = 1234, sum = 0;

while (num != 0) {

sum += num % 10;

num /= 10;

}

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

}

}

**34. // Find second largest in an array**

public class SecondLargest {

public static void main(String[] args) {

int[] arr = {12, 35, 1, 10, 34, 1};

int first = Integer.MIN\_VALUE, second = Integer.MIN\_VALUE;

for (int n : arr) {

if (n > first) {

second = first;

first = n;

} else if (n > second && n != first) {

second = n;

}

}

System.out.println("Second largest: " + second);

}

}

**35. // Count frequency of each character in string**

import java.util.\*;

public class CharFrequency {

public static void main(String[] args) {

String str = "hello world";

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

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

if (ch != ' ')

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

}

System.out.println(freq);

}

}

**36. // Remove duplicates from array**

import java.util.\*;

public class RemoveDuplicates {

public static void main(String[] args) {

int[] arr = {1, 2, 2, 3, 4, 4, 5};

Set<Integer> set = new LinkedHashSet<>();

for (int num : arr) set.add(num);

System.out.println(set);

}

}

**37. // Check leap year**

public class LeapYear {

public static void main(String[] args) {

int year = 2024;

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

System.out.println(isLeap ? "Leap Year" : "Not a Leap Year");

}

}

**38. // Reverse a string**

public class ReverseString {

public static void main(String[] args) {

String str = "hello";

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

System.out.println(reversed);

}

}

**39. // Convert string to integer**

public class StringToInt {

public static void main(String[] args) {

String str = "123";

int num = Integer.parseInt(str);

System.out.println(num);

}

}

**40. // Find smallest element in array**

public class MinArray {

public static void main(String[] args) {

int[] arr = {12, 3, 7, 0, -2};

int min = arr[0];

for (int num : arr) if (num < min) min = num;

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

}

}

**41. // Convert string to uppercase**

public class ToUpper {

public static void main(String[] args) {

String str = "hello";

System.out.println(str.toUpperCase());

}

}

**42. // Print Fibonacci using recursion**

public class FibonacciRecursion {

static int fib(int n) {

if (n <= 1) return n;

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

}

public static void main(String[] args) {

for (int i = 0; i < 10; i++) {

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

}

}

}

**43. // Sum of array elements**

public class SumArray {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

int sum = 0;

for (int num : arr) sum += num;

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

}

}

**44. // Program to check if two strings are anagrams**

import java.util.\*;

public class AnagramCheck {

public static void main(String[] args) {

String s1 = "listen", s2 = "silent";

char[] a = s1.toCharArray(), b = s2.toCharArray();

Arrays.sort(a); Arrays.sort(b);

System.out.println(Arrays.equals(a, b) ? "Anagram" : "Not Anagram");

}

}

**45. // Convert char to int and int to char**

public class CharIntConversion {

public static void main(String[] args) {

char c = 'A';

int i = c;

System.out.println("Char to Int: " + i);

i = 66;

c = (char)i;

System.out.println("Int to Char: " + c);

}

}

**46. // Check perfect number**

public class PerfectNumber {

public static void main(String[] args) {

int num = 28, sum = 0;

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

if (num % i == 0) sum += i;

}

System.out.println(sum == num ? "Perfect" : "Not Perfect");

}

}

**47. // Bubble sort**

public class BubbleSort {

public static void main(String[] args) {

int[] arr = {5, 1, 4, 2, 8};

for (int i = 0; i < arr.length - 1; i++) {

for (int j = 0; j < arr.length - 1 - i; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

System.out.println(Arrays.toString(arr));

}

}

**48. // Linear search**

public class LinearSearch {

public static void main(String[] args) {

int[] arr = {2, 4, 6, 8, 10};

int key = 6, index = -1;

for (int i = 0; i < arr.length; i++) {

if (arr[i] == key) { index = i; break; }

}

System.out.println(index == -1 ? "Not found" : "Found at index " + index);

}

}

**49. // Binary search**

public class BinarySearch {

public static void main(String[] args) {

int[] arr = {1, 3, 5, 7, 9};

int key = 7, low = 0, high = arr.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (arr[mid] == key) {

System.out.println("Found at index: " + mid);

return;

} else if (arr[mid] < key) low = mid + 1;

else high = mid - 1;

}

System.out.println("Not found");

}

}

**50. // Find factorial using loop**

public class FactorialLoop {

public static void main(String[] args) {

int n = 5, fact = 1;

for (int i = 1; i <= n; i++) fact \*= i;

System.out.println("Factorial: " + fact);

}

}

**Java Technical Questions**

1. **What are the main features of Java programming language?**
   * Answer: Java is simple, object-oriented, platform-independent (WORA), robust (exception handling), secure, architecture-neutral, portable, interpreted, high performance (JIT), multithreaded, and dynamic.
2. **Explain the concept of platform independence in Java.**
   * Answer: Java programs are compiled into bytecode, which runs on the JVM (Java Virtual Machine). JVMs exist for many platforms, so Java code can run anywhere.
3. **What are Java buzzwords?**
   * Answer: Key Java buzzwords include: Simple, Secure, Portable, Object-oriented, Robust, Multithreaded, Architecture-neutral, Interpreted, High-performance, and Dynamic.
4. **Explain data types in Java.**
   * Answer: Java has 8 primitive types: byte, short, int, long, float, double, char, and boolean. It also supports reference types like arrays and objects.
5. **Differentiate between int and Integer.**
   * Answer: int is a primitive type; Integer is a wrapper class that provides utility methods and can be used in collections.
6. **What is variable scope in Java?**
   * Answer: Scope defines the area where a variable is accessible: local, instance, and static.
7. **How does Java handle type conversion and casting?**
   * Answer: Java allows implicit conversion (widening) and explicit casting (narrowing). E.g., int a = (int) 5.7;
8. **Explain constants in Java.**
   * Answer: Constants are defined using the final keyword, e.g., final int MAX = 100;.
9. **What are control flow statements in Java?**
   * Answer: Java provides if, if-else, switch, for, while, do-while, break, continue.
10. **What is a constructor?**
    * Answer: A constructor initializes an object. It has the same name as the class and no return type.
11. **Explain the purpose of this keyword**.
    * Answer: this refers to the current object instance. It's used to differentiate class fields from method parameters.
12. **What is recursion?**
    * Answer: A function calling itself. Example: calculating factorial.
13. **What is inheritance in Java?**
    * Answer: Inheritance allows one class to acquire the properties and methods of another class using extends.
14. **Explain the use of super keyword.**
    * Answer: super refers to the parent class. Used to access parent class methods and constructors.
15. **What is method overloading and overriding?**
    * Answer:
      + Overloading: same method name, different parameters (compile time).
      + Overriding: same method in subclass (runtime polymorphism).
16. **What is an abstract class?**
    * Answer: A class with abstract methods. Cannot be instantiated. Used for base design.
17. **What is the difference between an abstract class and an interface?**
    * Answer:
      + Abstract class: can have constructor, instance variables.
      + Interface: only constants and abstract methods (Java 8+ allows static and default methods).
18. **What is the difference between final, finally, and finalize()?**
    * Answer:
      + final: constant or prevent overriding.
      + finally: block in exception handling.
      + finalize(): method for garbage collection (deprecated in modern Java).
19. **How do you prevent inheritance in Java?**
    * Answer: Use the final keyword in the class definition.
20. **What is an interface in Java?**
    * Answer: A contract that classes must implement. Defined using interface keyword.
21. **Can a class implement multiple interfaces?**
    * Answer: Yes, Java supports multiple inheritance via interfaces.
22. **What is an inner class in Java?**
    * Answer: A class defined within another class. Types: static, non-static, local, anonymous.
23. **What are access modifiers?**
    * Answer: Control visibility. Types: private, default, protected, public.
24. **What is constructor overloading?**
    * Answer: Defining multiple constructors with different parameter lists.
25. **What is exception handling in Java?**
    * Answer: Mechanism to handle runtime errors using try, catch, finally, throw, throws.
26. **Difference between checked and unchecked exceptions?**
    * Answer:
      + Checked: must be handled (IOException).
      + Unchecked: runtime errors (ArithmeticException).
27. **What is the use of finally block?**
    * Answer: Executes whether an exception is thrown or not. Used to release resources.
28. **How to create a custom exception?**
    * Answer: Extend the Exception class and override constructor.
29. **Explain multithreading.**
    * Answer: Process of executing multiple threads simultaneously. Improves efficiency.
30. **How to create a thread in Java?**
    * Answer: By extending Thread class or implementing Runnable interface.
31. **What is thread synchronization?**
    * Answer: Prevents race condition by controlling thread access using synchronized.
32. **What is the lifecycle of a thread?**
    * Answer: New → Runnable → Running → Blocked/Waiting → Terminated.
33. **What is the producer-consumer problem?**
    * Answer: Classic problem in synchronization, handled via wait() and notify().
34. **What is sleep() and yield() in threads?**
    * Answer:
      + sleep(): pause thread for a time.
      + yield(): pause to allow other threads of equal priority.
35. **What is the difference between start() and run() methods?**
    * Answer: start() creates a new thread and calls run(); calling run() directly runs it in the same thread.
36. **What is the Java Collections Framework?**
    * Answer: A set of classes and interfaces for storing and manipulating data (List, Set, Map, etc.).
37. **Difference between ArrayList and Vector?**
    * Answer:
      + ArrayList: not synchronized.
      + Vector: synchronized, slower.
38. **What is a Stack?**
    * Answer: LIFO data structure. Supports push, pop, peek.
39. **What is a Lambda expression?**
    * Answer: A concise way to write anonymous functions (Java 8+).
40. **Explain types of Java streams.**
    * Answer:
      + Byte streams: InputStream, OutputStream
      + Character streams: Reader, Writer
41. **What are file operations in Java?**
    * Answer: Use File class to create, delete, and get file information.
42. **Difference between text and binary files?**
    * Answer:
      + Text: human-readable.
      + Binary: stored in binary format, not human-readable.
43. **What is JDBC?**
    * Answer: Java Database Connectivity API. Connects Java apps to a database.
44. **Explain different types of JDBC drivers.**
    * Answer:
      + Type 1: JDBC-ODBC bridge
      + Type 2: Native API driver
      + Type 3: Network Protocol driver
      + Type 4: Thin driver (pure Java)
45. **What is AWT?**
    * Answer: Abstract Window Toolkit – Java's original GUI framework, platform dependent.
46. **What is Swing?**
    * Answer: Lightweight, platform-independent GUI toolkit. Part of Java Foundation Classes.
47. **Difference between AWT and Swing?**
    * Answer:
      + AWT: heavyweight, uses native resources.
      + Swing: lightweight, written in Java.
48. **What is a Layout Manager?**
    * Answer: Manages the layout of GUI components. E.g., FlowLayout, BorderLayout, GridLayout.
49. **What are Event Listeners in Java Swing?**
    * Answer: Interfaces that handle events like button clicks (ActionListener, MouseListener).
50. **What is the Delegation Event Model in Java?**
    * Answer: Separates event source from event handling logic using listener interfaces.