100 Comprehensive Programming Problems

Date: 2024
Difficulty Levels: Beginner to Advanced
Topics Covered: Basic Java, OOP, Data Structures, Algorithms, File Handling, Collections, Multithreading

---

1. Basic Java Programs (Questions 1-20)

1.1. Basic Input/Output and Operations

Question 1: Write a Java program that takes two numbers as input and displays their sum, difference, product, and quotient with proper exception handling.

Question 2: Create a program that reads a character and displays whether it's a vowel, consonant, digit, or special character.

Question 3: Implement a program that converts temperature between Celsius and Fahrenheit with menu-driven approach.

Question 4: Write a program to swap two numbers using a third variable and without using a third variable.

Question 5: Create a program that calculates simple and compound interest with user inputs.

1.2. Conditional Statements

Question 6: Write a program to find the largest among three numbers using if-else ladder.

Question 7: Implement a program that checks if a year is a leap year or not.

Question 8: Create a calculator using switch-case that handles basic arithmetic operations.

Question 9: Write a program that determines the type of triangle based on sides (equilateral, isosceles, scalene).

Question 10: Implement a program that calculates electricity bill based on units consumed with slab rates.

1.3. Looping Structures

Question 11: Write a program to print all prime numbers between 1 and 100.

Question 12: Create a program to find the sum of all even numbers between 1 and n.

Question 13: Implement a program to print various patterns (pyramid, diamond, number patterns).

Question 14: Write a program to find the GCD and LCM of two numbers using Euclidean algorithm.

Question 15: Create a program that reverses a number and checks if it's a palindrome.

Question 16: Implement a program to find all Armstrong numbers between 1 and 1000.

Question 17: Write a program to calculate the sum of series: 1 + 1/2 + 1/3 + ... + 1/n.

Question 18: Create a program to convert decimal to binary, octal, and hexadecimal.

Question 19: Implement a program to find the factorial of a number using recursion.

Question 20: Write a program to generate Fibonacci series up to n terms using both iteration and recursion.

---

2. Methods and Arrays (Questions 21-35)

2.1. Methods/Functions

Question 21: Write a method that checks if a number is prime and use it to find all prime numbers in a range.

Question 22: Create a method to calculate power of a number using recursion.

Question 23: Implement a method that returns the reverse of a number.

Question 24: Write a method overloading example with different parameter types.

Question 25: Create a recursive method to find the sum of digits of a number.

2.2. Arrays

Question 26: Write a program to find the largest and smallest element in an array.

Question 27: Implement a program to sort an array using bubble sort algorithm.

Question 28: Create a program to search an element in an array using linear and binary search.

Question 29: Write a program to merge two sorted arrays into one sorted array.

Question 30: Implement a program to find the frequency of each element in an array.

Question 31: Create a program to rotate an array by k positions.

Question 32: Write a program to find the second largest element in an array.

Question 33: Implement a program to remove duplicates from a sorted array.

Question 34: Create a program to find the union and intersection of two arrays.

Question 35: Write a program to implement matrix addition, subtraction, and multiplication.

---

3. Object-Oriented Programming (Questions 36-55)

3.1. Classes and Objects

Question 36: Create a Student class with attributes like name, roll number, marks, and methods to calculate grade and display information.

Question 37: Implement a Bank Account class with deposit, withdraw, and balance inquiry functionality.

Question 38: Write a Circle class with methods to calculate area, circumference, and diameter.

Question 39: Create a Rectangle class with methods to calculate area, perimeter, and diagonal.

Question 40: Implement a Complex Number class with methods for arithmetic operations.

3.2. Constructors

Question 41: Create a Book class with appropriate constructors (default, parameterized, copy) and demonstrate their usage.

Question 42: Implement a Date class with validation for day, month, year and methods to compare dates.

Question 43: Write a Time class with methods for addition and subtraction of time.

Question 44: Create a String manipulation class that mimics basic string operations.

Question 45: Implement a Stack class using arrays with push, pop, and peek operations.

3.3. Inheritance

Question 46: Create a base class Shape with derived classes Circle, Rectangle, Triangle implementing area calculation.

Question 47: Implement a Person class with Student and Teacher as derived classes.

Question 48: Write a Vehicle class with Car and Bike as derived classes, each with specific attributes.

Question 49: Create a base class Employee with derived classes FullTime and PartTime with different salary calculations.

Question 50: Implement multilevel inheritance: Person → Student → GraduateStudent.

3.4. Polymorphism and Abstraction

Question 51: Create abstract base class Animal with derived classes Dog, Cat implementing abstract methods.

Question 52: Implement method overloading and overriding with practical examples.

Question 53: Write a program demonstrating runtime polymorphism with method overriding.

Question 54: Create an interface for different payment methods (CreditCard, PayPal, etc.).

Question 55: Implement multiple inheritance using interfaces.

---

4. Exception Handling (Questions 56-65)

Question 56: Write a program to handle ArithmeticException for division by zero.

Question 57: Implement custom exception classes for banking application (InsufficientFundsException).

Question 58: Create a program that handles ArrayIndexOutOfBoundsException.

Question 59: Write a program demonstrating try-catch-finally block with multiple catch blocks.

Question 60: Implement exception handling for file operations (FileNotFoundException, IOException).

Question 61: Create a program that uses throw and throws keywords appropriately.

Question 62: Write a program to handle NumberFormatException for invalid number conversions.

Question 63: Implement a program with nested try-catch blocks.

Question 64: Create a custom exception for age validation (InvalidAgeException).

Question 65: Write a program that demonstrates exception propagation.

---

5. Java Collections Framework (Questions 66-80)

5.1. List Interface

Question 66: Write a program to demonstrate ArrayList operations (add, remove, iterate, sort).

Question 67: Implement a program using LinkedList for student record management.

Question 68: Create a program to compare ArrayList and LinkedList performance.

Question 69: Write a program to remove duplicates from an ArrayList.

Question 70: Implement a program to sort ArrayList of custom objects.

5.2. Set Interface

Question 71: Create a program using HashSet to store unique elements.

Question 72: Implement a program using TreeSet for sorted unique elements.

Question 73: Write a program to find common elements between two sets.

Question 74: Create a program demonstrating LinkedHashSet for maintaining insertion order.

Question 75: Implement a program to remove duplicates using Set.

5.3. Map Interface

Question 76: Write a program using HashMap to count word frequency in a text.

Question 77: Create a program using TreeMap for sorted key-value pairs.

Question 78: Implement a phone directory using HashMap.

Question 79: Write a program to iterate through Map using different methods.

Question 80: Create a program demonstrating LinkedHashMap for maintaining insertion order.

---

6. File Handling and I/O (Questions 81-90)

Question 81: Write a program to create a file and write student information to it.

Question 82: Implement a program to read from a file and display its contents.

Question 83: Create a program to copy contents from one file to another.

Question 84: Write a program to count words, lines, and characters in a file.

Question 85: Implement a student record system using file handling (add, search, delete records).

Question 86: Create a program to merge two files into a third file.

Question 87: Write a program to find and replace specific text in a file.

Question 88: Implement a program that handles binary files for storing objects.

Question 89: Create a program to maintain inventory using file handling.

Question 90: Write a program that demonstrates serialization and deserialization.

---

7. Multithreading (Questions 91-95)

Question 91: Create a program that demonstrates thread creation by extending Thread class.

Question 92: Implement a program that creates threads by implementing Runnable interface.

Question 93: Write a program demonstrating thread synchronization for bank account operations.

Question 94: Create a producer-consumer problem solution using wait() and notify().

Question 95: Implement a program demonstrating thread priorities and sleep method.

---

8. Advanced Java Concepts (Questions 96-100)

Question 96: Write a program using Java 8 features (Lambda expressions, Stream API).

Question 97: Create a program demonstrating Optional class to avoid NullPointerException.

Question 98: Implement a program using Java Date and Time API (LocalDate, LocalTime, LocalDateTime).

Question 99: Write a program using Java Networking (Socket programming) for client-server communication.

Question 100: Implement a complete banking application with all OOP concepts, collections, exception handling, file handling, and multithreading.