Problem Statement 1:

Customer Account Management System with Database Integration

Overview:
Develop a **Customer Account Management System** in Java that allows bank employees to manage customer accounts using a collection (List) and a relational database (Oracle 11g/PostgreSQL). The system should allow adding new customer accounts and searching by customer name.

Requirements

1. Class Design & Package Structure

Customer (com.model)

- Represents a customer with the following attributes:
    - customerId (int)
    - name (String)
    - accountType (String) – e.g., "Savings", "Current"
    - balance (double)
- Implements:
    - Parameterized constructor to initialize a Customer object.
    - toString() method to display customer details.

DBConnection (com.utility)

- Static getConnection() method to establish a connection to the database (Oracle/PostgreSQL).

BankStore (com.store)

- Manages customer records using both an in-memory collection (List) and a database.
- Provides the following methods:
    - void addCustomers(List<Customer> customers)
      Adds a list of customers to the database.

- List<Customer> searchByName(String name)
  Searches customers matching the name, adds them to a list, and returns the list.

BankUtil (com.utils - Main class)

- Create a few Customer objects (including multiple customers with the same name) and add them to a list.
- Invoke addCustomers() method by passing the list.
- Accept customer name from user input.
- Call searchByName() and print all customers matching the name.

Marks Evaluation Rubrics: Total: 60 Marks

Class Design and Implementation (15 points)	Class Structure (10 points)
	Parameterized Constructor (5 points)
Collection and Methods Implementation (25 points)	Collection Usage (10 points)
	Method implementation (15)
Output & Coding Practices (20 points)	Output(15)
	Clean coding practice(5)

Problem Statement 2:

Implementing Dependency Injection in a Banking Application (Spring Core)

You are developing a **Basic Banking Application** using **Spring Core**. The application should support basic banking operations like checking balance and performing withdrawals.

Requirements

1. Create a BankingService (com.service package)

This class provides business logic methods:

- double checkBalance(int accountNumber) – Returns a dummy balance.
- double withdraw(int accountNumber, double amount) – Returns balance after withdrawal (dummy logic).

2. Create a BankController (com.controller package)

- This class depends on BankingService.
- It should call the service methods for balance checking and withdrawal.

3. Use Spring Dependency Injection (DI)

- Inject BankingService into BankController using **Annotation-based configuration** (@Component, @Autowired) **or** XML-based configuration.

4. Create a Main class App (com.utils package)

- Use ApplicationContext to retrieve the BankController bean.
- Call both operations (checkBalance() and withdraw()) and display results.

Marks Evaluation Rubrics: Total: 40 Marks

Criteria	Marks
Class Design and Business Logic	10
Spring Configuration (Annotations/XML)	10
Defining Main Class with Context & Output	5
Output as per requirements	10
Clean coding (naming, structure, comments)	5
Total	**40 Marks**

Naming & Submission Guidelines:

- Compress each project folder separately into a ZIP file.
- Name each ZIP file using the following format:
  - **yourname_problemstatement_number.zip**
- Replace yourname with your actual name.

- Replace problemstatement_number with the specific problem number assigned to you.

- Example:
  If your name is Ravi Kumar, your ZIP files should be named as:

  ravikumar_problemstatement_01.zip
  ravikumar_problemstatement_02.zip