Consider a simple bank account system where you have customers and their bank accounts. Each customer has a name, a unique identifier, and can have one or more bank accounts. Each bank account has an account number, a balance, and belongs to a specific customer.Design a set of Java classes to model this bank account system. Your implementation should include the following features.

Customer Class:

- Attributes: name, customerID, bankAccount
- Methods: a method to add a bank account, a method to display customer information, methods to deposit and withdraw money

BankAccount Class:

- Attributes: accountNumber, balance
- Methods: a method to deposit money, a method to withdraw money, a method to display account information.

1. Create customer and bankAccount objects. And only using the customer object, deposit or withdraw money.

2. Write a Java program to create a class named `Book` that represents basic information about a book in a library. The class should have three attributes: `title` (String), `author` (String), and `yearPublished` (int). Include a constructor to initialize these fields and a method called `displayInfo()` that prints the book's details. In the `main` method, create at least two objects of the `Book` class with different data and call the `displayInfo()` method for each to display their information.

3.Design a `MetroTicket` class to simulate a metro ticket booking system using **constructor overloading**. Passengers can book tickets with different levels of information: some may provide only their name, source, and destination; others may include whether it's a round trip, choose between "Regular" or "Premium" seat classes, and even apply discount codes. The class should have four overloaded constructors to handle these scenarios. Use constructor chaining to avoid code repetition. Include validations—for instance, seat class must be either "Regular" or "Premium", and if an invalid value is given, default to "Regular". Implement methods like `calculateFare()` to compute the fare based on trip type and seat class (e.g., one-way Regular = 100, Premium = 150; round-trip Regular = 180, Premium = 270), with a 10% discount if the code "DIS10" is applied. Also, write a `displayTicketDetails()` method to show all ticket information. Create at least four objects using different constructors to demonstrate constructor overloading effectively.

4. A department registers students for a course using a fixed-size
system. Each student has a **student ID**, **name**, and **CGPA**.
The administrator can:

- Register a new student
- Search a student by ID
- Display all registered students

Task

Implement the system using an **array of Student objects**.