Collection Problems

- Employee Management System:

- Create a class Employee with attributes id, name, and salary. Write a program to add multiple Employee objects to an ArrayList, display all employees, and calculate the average salary of all employees.

- Student Marks Tracker:

- Define a Student class with attributes rollNo, name, and a list of marks for 5 subjects. Store multiple Student objects in an ArrayList and write methods to:

  - Display each student’s details along with their average marks.
  - Find and display the topper (student with the highest average marks).

- Product Catalog:

- Implement a class Product with attributes productId, productName, and price. Create an ArrayList to store products and implement functionalities to:

  - Add new products to the list.
  - Find and display products cheaper than a user-specified price.
  - Sort the list of products by price in ascending order.

- Library Management System:

- Create a Book class with attributes bookId, title, author, and availabilityStatus. Use an ArrayList to store multiple books and provide functionalities to:

  - Search for a book by title or author.
  - Borrow a book (mark it as unavailable).Display all available books.

- Cinema Booking System:

- Create a Movie class with attributes movieId, movieName, genre, and rating. Store multiple Movie objects in an ArrayList and implement methods to:

  - Display all movies of a specific genre.
  - Find the highest-rated movie.
  - Sort movies by rating in descending order.

- Unique Students in a Class

- Create a Student class with attributes id, name, and grade. Store multiple Student objects in a Set and write a program to ensure that no duplicate students (based on id) are added. Use HashSet and override equals() and hashCode().

- Library Book Tracker

- Create a Book class with attributes isbn, title, and author. Use a Set to store the books available in a library. Implement functionality to:

  - Add a book to the library (prevent duplicates based on isbn).
  - Check if a specific book (by isbn) is available in the library.

- Unique Employee Departments

- Create an Employee class with attributes empId, name, and department. Use a TreeSet to store employees sorted by department name. Implement functionalities to:

  - Add employees to the set.
  - Display all employees sorted by department

- Duplicate Removal from a List of Products

- Define a Product class with attributes productId, name, and price. Given a list of Product objects, write a program to remove duplicate products (based on productId) and store the unique products in a HashSet.

- Student-Grades Tracker

- Create a Student class with attributes rollNumber and name. Use a Map where the key is the rollNumber and the value is the Student object. Implement functionalities to:

  - Add new students to the map.
  - Retrieve a student by their roll number.
  - Display all students in the map.

- Student Attendance Tracker

- Create a Student class with attributes rollNumber and name. Simulate a system where the attendance of students is marked in a LinkedHashSet to preserve the order of attendance. Implement methods to:

  - Mark attendance for a student.
  - Display the attendance list in the order it was marked.

- Employee-Salary Management

- Create an Employee class with attributes empId, name, and department. Use a Map where the key is the empId and the value is the Employee object. Write a program to:

  - Add employees to the map.
  - Update the salary of an employee by their empId.
  - Display all employees in a specific department.

- Library Book Management

- Create a Book class with attributes isbn, title, and author. Use a Map where the key is the isbn and the value is the Book object. Implement methods to:

    - Add a book to the library.
    - Search for a book by its isbn.
    - Display all books in the library.

- Inventory System

- Create a Product class with attributes productId, name, and quantity. Use a Map where the key is the productId and the value is the Product object. Write a program to:

  - Add products to the inventory.
  - Update the quantity of a product by its productId.
  - Display all products with a quantity below a certain threshold.

- Order Management System

- Create an Order class with attributes orderId, customerName, and amount. Use a Map where the key is the orderId and the value is the Order object. Write a program to:

  - Add orders to the system.
  - Cancel an order by its orderId.
  - Display the total value of all orders in the system.