Java Collection Assignments with Custom Data Types

Assignment 1: Employee Directory (Using Set)

Scenario:
Your company needs an **Employee Directory** that ensures no duplicate employees.

Custom Data Type:
Employee { int id, String name, String department }

Requirements:

1. Use a **HashSet<Employee>** to store employees (override equals() and hashCode() to avoid duplicates).
2. Implement CRUD operations:
    - Add a new employee.
    - Remove an employee by ID.
    - Update an employee's department.
    - View all employees.
3. Searching:
    - Find an employee by ID.
    - Find all employees in a given department.
4. Sorting:
    - Display employees sorted by name (Comparator).
    - Display employees sorted by department and then by ID.

👉 **Learning Outcome:** Prevent duplicates, perform search, and custom sorting on complex objects.

Assignment 2: Product Price Catalog (Using Map)

Scenario:
You are designing a **Product Price Catalog** for an e-commerce system.

Custom Data Type:
Product { int productId, String name, double price }

Requirements:

1. Store products using a **HashMap<Integer, Product>** where:
    - Key = productId
    - Value = Product object

2. CRUD operations:
    - Add a new product.
    - Remove a product.
    - Update the price of a product.
    - View all products.
3. Searching:
    - Find a product by its ID.
    - Find all products cheaper than a given price.
4. Sorting:
    - Display products sorted by product name.
    - Display products sorted by price (ascending/descending).

👉 **Learning Outcome:** Learn key-value storage, searching, and sorting maps with custom objects.

Assignment 3: Student Course Enrollment (Combined Set + Map)

Scenario:
A **University Enrollment System** is being developed to manage student registrations.

Custom Data Types:

- Course { int courseId, String courseName }
- Student { int studentId, String name }

Requirements:

1. Store all available courses in a **HashSet<Course>** (no duplicates).
2. Maintain a **Map<Student, Set<Course>>** to store student registrations.
    - Key = Student object
    - Value = Set of registered courses
3. CRUD operations:
    - Add a new course.
    - Register a student for one or more courses.
    - Update a student's registration (drop/add courses).
    - Remove a student from the system.
4. Searching:
    - Find all courses taken by a student.

    - Find all students registered for a given course.
5. Sorting:
    - Display all courses sorted by courseName.
    - Display all students sorted by name.

👉 **Learning Outcome:** Combination of Set and Map with custom objects, handling many-to-many relationships.

Assignment 4: Vehicle Parking Management System (Real-time Set + Map)

Scenario:
A shopping mall needs a **Parking Management System**.

Custom Data Types:

- Vehicle { String plateNumber, String ownerName, String type }
- ParkingSlot { int slotId, String location }

Requirements:

1. Use a **HashSet<Vehicle>** to ensure no duplicate vehicles are parked.
2. Maintain a **Map<ParkingSlot, Vehicle>** to track which slot has which vehicle.
3. CRUD operations:
    - Park a vehicle in a slot.
    - Remove a vehicle when it exits.
    - Update vehicle details.
    - View all occupied slots.
4. Searching:
    - Find which slot a given vehicle is parked in.
    - Find all vehicles of type “SUV” or “Bike”.
5. Sorting:
    - Display all vehicles sorted by owner’s name.
    - Display all slots sorted by slotId.

👉 **Learning Outcome:** Real-world mapping of slots to vehicles, uniqueness with Set, search & sorting of custom objects.

✅ With these **4 industry-oriented assignments**:

- Get exposure to **Set** (uniqueness), **Map** (key-value), and **Combined Scenarios** (many-to-many).
- Also practice **CRUD, searching, sorting, and overriding equals/hashCode** in **custom data types**.
- Each problem reflects **real industry use-cases** like Employee Management, E-Commerce, University Enrollment, and Parking Systems.