C-DAC Mumbai
OOPJ Lab Assignment

Implementation Guidelines
For Solving Each Problem:

1. **Choose appropriate collection type** based on requirements
2. **Implement proper error handling** for edge cases
3. **Use generics** where applicable for type safety
4. **Follow Java naming conventions**
5. **Include proper documentation** and comments

Key Learning Outcomes:

- Understanding when to use different collection types
- Mastering iteration techniques and removal
- Working with generics and custom objects
- Implementing real-world scenarios using collections
- **During Interview you can mention these use-cases for a particular concept**

Problem 1: Student Names Management System
Use Case: A school administrator needs to maintain a list of student names for a class roster.
Requirements:

- Add student names to the roster
- Display all students
- Remove a student from the roster

Sample Input:
Add students: "Amit", "Priya", "Rohan"
Remove student: "Priya"

Expected Output:
Students: Amit, Rohan

Problem 2: Lab Access Queue System
Use Case: A computer lab needs to manage students waiting for access using a first-come-first-served system.
Requirements:

- Students join the queue for lab access
- Process students in FIFO order
- Display remaining queue

Sample Input:
Enqueue: "Amit", "Priya", "Rohan"
Dequeue: 1 student

Expected Output:
Queue: Priya, Rohan

C-DAC Mumbai
OOPJ Lab Assignment

Problem 3: Daily Task Tracker
Use Case: A student wants to track their daily tasks and mark completed ones.
Requirements:

- Add tasks to the list
- Mark tasks as completed (remove them)
- Display remaining tasks

Sample Input:
Add tasks: "Study Java", "Complete Assignment", "Exercise"
Complete task: "Exercise"

Expected Output:
Remaining tasks: Study Java, Complete Assignment

Problem 4: Grocery Shopping List
Use Case: A person maintains a grocery list and removes items as they purchase them.
Requirements:

- Add items to grocery list
- Remove purchased items
- Display remaining items

Sample Input:
Add items: "Milk", "Eggs", "Bread"
Purchase: "Milk"

Expected Output:
Items to buy: Eggs, Bread

Problem 5: Recent Search History
Use Case: A search application maintains the last 5 searches, removing the oldest when the limit is exceeded.
Requirements:

- Store recent searches (maximum 5)
- Remove oldest search when limit exceeded
- Maintain insertion order

Sample Input:
Searches: "Java", "Python", "C++", "DSA", "OOP", "Spring"

Expected Output:
Recent searches: Python, C++, DSA, OOP, Spring

C-DAC Mumbai
OOPJ Lab Assignment

Problem 6: Unique Roll Number Validator
Use Case: A registration system must ensure no duplicate roll numbers are assigned.
Requirements:

- Accept roll numbers for registration
- Automatically remove duplicates
- Display unique roll numbers

Sample Input:
Roll numbers: 101, 102, 101, 103

Expected Output:
Unique Roll Numbers: 101, 102, 103

Problem 7: Alphabetical Student Directory
Use Case: A school wants to maintain student names in alphabetical order.
Requirements:

- Add student names
- Automatically maintain alphabetical sorting
- Display sorted name

Sample Input:
Students: "Rohan", "Amit", "Priya"

Expected Output:
Students: Amit, Priya, Rohan

Problem 8: Course Registration System
Use Case: A student registers for courses, ensuring no duplicate course codes.
Requirements:

- Register for courses using course codes
- Prevent duplicate registrations
- Display registered courses

Sample Input:
Course codes: "CS101", "MA101", "CS101"

Expected Output:
Registered Courses: CS101, MA101

C-DAC Mumbai
OOPJ Lab Assignment

Problem 9: Event Attendance Counter
Use Case: Count unique attendees at an event, handling duplicate check-ins.
Requirements:

- Record attendee names
- Count only unique attendees
- Handle duplicate entries

Sample Input:
Attendees: "Amit", "Rohan", "Amit", "Priya"

Expected Output:
Total unique attendees: 3

Problem 10: Electronic Voting System
Use Case: Track unique voters in an election system to prevent duplicate voting.
Requirements:

- Record voter IDs
- Ensure one vote per voter
- Count total unique voters

Sample Input:
Voter IDs: 201, 202, 203, 202

Expected Output:
Total voters: 3

Problem 11: Student Grade Management
Use Case: A teacher needs to map student names to their exam marks.
Requirements:

- Store student name and marks pairs
- Retrieve marks by student name
- Display all student-marks mappings

Sample Input:
Students and marks: "Amit" → 85, "Priya" → 92, "Rohan" → 78

Expected Output:
Grade Report: Amit:85, Priya:92, Rohan:78

C-DAC Mumbai
OOPJ Lab Assignment

Problem 12: Attendance Tracking System
Use Case: Track student attendance percentages in alphabetical order.
Requirements:

- Map student names to attendance percentages
- Maintain alphabetical order of students
- Display sorted attendance report

Sample Input:
Attendance: "Amit" → 90, "Rohan" → 85, "Priya" → 95

Expected Output:
Attendance Report: Amit:90, Priya:95, Rohan:85

Problem 13: Student Registration Order Tracker
Use Case: Maintain the order in which students registered for a course.
Requirements:

- Record registration order
- Map student names to roll numbers
- Preserve insertion order

Sample Input:
Registrations: "Amit" → 101, "Rohan" → 102, "Priya" → 103

Expected Output:
Registration Order: Amit:101, Rohan:102, Priya:103

Problem 14: Grade Update System
Use Case: Update a student's marks in the grading system.
Requirements:

- Store student grades
- Update existing student's marks
- Display updated information

Sample Input:
Initial: "Rohan" → 78
Update: "Rohan" → 88

Expected Output:
Updated Grade: Rohan:88

Problem 15: Library Book Inventory
Use Case: Track available copies of books in a library system.
Requirements:

- Map book titles to available copies
- Update copies when books are borrowed
- Display current inventory

Sample Input:
Initial inventory: "Java" → 3, "Python" → 5
Borrow: "Java" (1 copy)

Expected Output:
Current inventory: Java:2, Python:5

Problem 16: Grade-Based Student Filter
Use Case: Remove students with failing grades from the honor roll.
Requirements:

- Store student grades in a map
- Remove students with marks below 60
- Display remaining student

Sample Input:
Student grades: "Amit" → 85, "Priya" → 52, "Rohan" → 78
Filter threshold: 60

Expected Output:
Honor Roll: Amit:85, Rohan:78

Problem 17: Grade Distribution Counter
Use Case: Analyze the distribution of grades in a class.
Requirements:

- Count frequency of each grade
- Display grade distribution
- Handle multiple occurrences

Sample Input:
Grades: ["A", "B", "A", "C", "B", "A"]

Expected Output:
Grade Distribution: A=3, B=2, C=1

Problem 18: Batch Merger System
Use Case: Merge student lists from morning and evening batches, removing duplicates.
Requirements:

- Combine two student lists
- Remove duplicate students
- Maintain unique student list

Sample Input:
Morning batch: "Amit", "Priya"
Evening batch: "Rohan", "Priya"

Expected Output:
Combined batches: Amit, Priya, Rohan

Problem 19: Grade Report Generator
Use Case: Display all student grades using proper iteration techniques.
Requirements:

- Iterate through student-grade mappings
- Display formatted grade report
- Use Iterator pattern

Sample Input:
Student grades: "Amit" → 85, "Priya" → 92

Expected Output:
Grade Report: Amit:85, Priya:92

Problem 20: Even Roll Number Filter
Use Case: Filter and display only students with even roll numbers.
Requirements:

- Process list of roll numbers
- Remove odd roll numbers
- Display filtered results

Sample Input:
Roll numbers: 101, 102, 103, 104

Expected Output:
Even Roll Numbers: 102, 104

C-DAC Mumbai
OOPJ Lab Assignment

Problem 21: Text Editor Undo System
Use Case: Implement undo functionality for a text editor.
Requirements:

- Track user actions in order
- Implement undo operation (LIFO)
- Display current action history

Sample Input:
Actions: "Type A", "Type B", "Delete"
Operation: Undo last action

Expected Output:
Current actions: Type A, Type B

Problem 22: Ticket Booking Queue
Use Case: Manage customer service in a ticket booking system.
Requirements:

- Queue customers for service
- Serve customers in FIFO order
- Display current queue status

Sample Input:
Queue: "Amit", "Priya", "Rohan"
Serve: 1 customer

Expected Output:
Serving: Amit, Queue: Priya, Rohan

Problem 23: Browser History Management
Use Case: Maintain browser history with back functionality.
Requirements:

- Store visited pages
- Implement back navigation (LIFO)
- Display current history

Sample Input:
Pages visited: "Google", "YouTube", "GFG"
Action: Back (1 page)

Expected Output:
Current history: Google, YouTube

C-DAC Mumbai
OOPJ Lab Assignment

Problem 24: Print Job Queue Manager
Use Case: Manage print jobs in a shared printer system.
Requirements:

- Queue print jobs
- Process jobs in order
- Display job status

Sample Input:
Jobs: "Doc1", "Doc2", "Doc3"
Process: 1 job

Expected Output:
Printing Doc1, Queue: Doc2, Doc3

Problem 25: Command History Tracker
Use Case: Store recent commands in a terminal with limited history.
Requirements:

- Maintain last 3 commands
- Remove oldest when limit exceeded
- Display recent commands

Sample Input:
Commands: "ls", "pwd", "cd ..", "mkdir"
History limit: 3

Expected Output:
Recent Commands: pwd, cd .., mkdir

Problem 26: Employee Management System
Use Case: Manage employee information including name and salary.
Requirements:

- Create Employee objects with name and salary
- Store employees in a collection
- Display employee information

Sample Input:
Employees: Employee("Amit", 50000), Employee("Priya", 60000)

Expected Output:
Employee List: Amit:50000, Priya:60000

C-DAC Mumbai
OOPJ Lab Assignment

Problem 27: Employee Salary Sorting
Use Case: Sort employees by salary for payroll processing.
Requirements:

- Store employee objects
- Sort by salary in ascending order
- Display sorted employee list

Sample Input:
Employees: ("Amit", 50000), ("Priya", 60000), ("Rohan", 45000)

Expected Output:
Sorted by salary: Rohan:45000, Amit:50000, Priya:60000

Problem 28: Department Employee Mapping
Use Case: Organize employees by department for HR management.
Requirements:

- Map departments to employee lists
- Group employees by department
- Display departmental structure

Sample Input:
Department mapping: "IT" → ["Amit", "Rohan"], "HR" → ["Priya"]

Expected Output:
Department Structure: IT: Amit, Rohan; HR: Priya

Problem 29: Student Record System
Use Case: Maintain student records with name and grade information.
Requirements:

- Create Student objects with name and grade
- Store in a collection
- Display student records

Sample Input:
Students: Student("Amit", "A"), Student("Priya", "B")

Expected Output:
Student Records: Amit:A, Priya:B

C-DAC Mumbai
OOPJ Lab Assignment

Problem 30: Grade-Based Student Filter
Use Case: Filter students based on minimum grade requirements.
Requirements:

- Store student objects with grades
- Remove students below grade B
- Display filtered results

Sample Input:
Students: ("Amit", "A"), ("Priya", "C"), ("Rohan", "B")
Filter: Grade >= B

Expected Output:
Qualified Students: Amit:A, Rohan:B

University Student Management System

Placement Pakka Problem Statement:

Create a **University Student Management System** using Java Collections to manage students across departments.

- **Roll Number** (int)
- **Name** (String)
- **Department** (String)
- **CGPA** (double)

Requirements

1. Registration List - ArrayList<Student>
- Store students in registration order

2. Merit List - Comparable<Student>
- Sort by CGPA (descending order)

3. Alphabetical List - Comparator<Student>
- Sort by name (A to Z)

4. Department Grouping - HashMap<String, List<Student>>
- Group students by department

5. Unique Names - HashSet<String>
- Track unique student names

6. Roll Number Sorting - TreeSet<Student>
- Auto-sort by roll number

7. Performance Filter - Iterator
- Remove students with CGPA < 5.0

8. Recent Registrations - Stack<Student>
- Track last added students (LIFO)

9. Scholarship Queue - Queue<Student>
- Process students for scholarships (FIFO)

10. Hostel Applications - LinkedList<Integer>
- Add priority applicants at **front**
- Add regular applicants at **end**
- Remove from **both ends** for allocation

Sample Data
Student s1 = new Student(101, "Amit", "CS", 8.5);
Student s2 = new Student(102, "Priya", "Math", 9.2);
Student s3 = new Student(103, "Rohan", "CS", 7.8);
Student s4 = new Student(104, "Sneha", "Physics", 4.5);

Expected Outputs

Registration Order: Amit, Priya, Rohan, Sneha
Merit List: Priya(9.2), Amit(8.5), Rohan(7.8), Sneha(4.5)
Alphabetical: Amit, Priya, Rohan, Sneha
Department Groups:
CS: [Amit, Rohan]
Math: [Priya]
Physics: [Sneha]

After Filter (CGPA >= 5.0): Amit, Priya, Rohan

Hostel Queue:
Add regular(105): [105]
Add priority(101): [101, 105]
Remove front: [105]

CONGRATULATIONS
