HashSet, LinkedHashSet & TreeSet

Section A: Basic Implementation

1. **Create a `HashSet` of strings** to store country names.
    - Add at least 5 country names, including a duplicate.
    - Display the set.
    - Explain the output order and behavior.
2. **Implement a `LinkedHashSet`** to store student roll numbers.
    - Insert roll numbers in a random order with some duplicates.
    - Display the output and explain how the order differs from `HashSet`.
3. **Use `TreeSet` to store integers** (e.g., scores of a game).
    - Insert unsorted and duplicate values.
    - Display the sorted result.
    - What happens when a `null` is added?

Section B: Functional Assignments

4. **Write a Java program to remove duplicates from a list of employee names** using:
    - `HashSet`
    - `LinkedHashSet` (to preserve insertion order)
5. **Create a program that accepts city names from the user until they type `exit`.**
    - Store the city names in a `TreeSet`.
    - Display the sorted list of unique cities.
6. **Design a program to compare the performance of `HashSet`, `LinkedHashSet`, and `TreeSet`**
    - Insert 10,000 random integers in each set.
    - Measure and display the time taken for:
        - Insertion
        - Search (e.g., check if 5000 is present)
        - Deletion

Comprehensive Problem

Problem Statement:

Create a **Student Management System** in Java using Object-Oriented Programming principles. The system will store unique student records in a Set collection.

Requirements:

1. Create an **interface** `PersonInfo` with:

   ```
   void displayInfo();
   ```

2. Create a **class** `Student` that implements `PersonInfo` and contains:
   - Fields: `int id`, `String name`, `double percentage`
   - A **parameterized constructor**
   - An overridden `toString()` method to display student details
   - Implementation of `displayInfo()` that prints the student using `toString()`
3. In the `main` method:
   - Ask the user to choose the type of Set to use:
     - `HashSet` (no ordering)
     - `LinkedHashSet` (insertion order)
     - `TreeSet` (will throw exception — intentional)
   - Ask how many student records to enter
   - Accept `id`, `name`, and `percentage` for each student and store each in the Set
   - Display all student details

Expected Behavior:

- **Duplicates allowed** if data is same but object references differ
- If **TreeSet** is used, a `ClassCastException` will occur —explore **why?**