Exercise: Build the Employee Management Project

Task 1: Create an Employee Class

1. Design a class to represent an employee with name, age, and salary details.
2. Add methods to compare employees and display their details.
3. Write unit tests to ensure:
    - Employees can be compared correctly (e.g., by salary).
    - Employee details are displayed in a consistent format.

Task 2: Develop an Employee Manager

1. Create a class to manage a collection of employees.
2. Implement functionality to:
    - Sort employees based on different criteria.
    - Group employees into categories.
    - Identify specific subsets of employees (e.g., highest earners).
    - Load employee data from a file.
    - Save employee data to a file.
3. Write tests to verify:
    - Sorting works as expected.
    - Grouping produces the correct categories.
    - File reading and writing handle various scenarios (e.g., empty or malformed files).

Task 3: Build the Main Application

1. Create a program that utilizes the employee-manager to:
    - Load employees from a file.
    - Perform operations like sorting and grouping.
    - Display results to the user.
    - Save modified data to a file.
2. Test the program with sample data to ensure all features work as expected.

Task 4: Test and Debug

1. Write comprehensive unit tests for all implemented functionality.
2. Run the program with various inputs (e.g., different file formats, missing data) to check its robustness.
3. Fix any identified bugs and optimize the program as needed.

Task 5: Enhance the Application (Optional)

1. Add additional features such as filtering or searching employees based on criteria.
2. Improve error handling and provide meaningful messages for incorrect inputs.
3. Make the program interactive by allowing users to select operations dynamically.

General Guidelines for Tests

1. **Class-Level Tests**
    - Ensure each class behaves as expected individually.
    - Focus on the core responsibilities of the class.

Tests for the Employee Class:

- Verify that employees can be compared for sorting (e.g., by salary).
- Ensure the employee details are represented in a consistent string format.

Tests for the EmployeeManager Class:

- **Sorting**:
    - Validate that employees can be sorted by different attributes (e.g., name, age).
- **Grouping**:
    - Confirm that employees are grouped into appropriate categories (e.g., age ranges).
- **File Operations**:
    - Test loading employee data from a file, ensuring it handles headers, empty lines, and malformed rows.
    - Test saving employee data to a file, validating the format and content.
- **Subset Extraction**:
    - Ensure methods for retrieving subsets (e.g., top earners) return the correct results.

Goal of Tests

- Ensure correctness, robustness, and reliability of functionality.
- Provide confidence in the system's behavior under various conditions.

Once you complete the project, please send it to my email: malik.krijestorac@stu.ssst.edu.ba