Exercise: The 'S' and 'O' in the SOLID acronym

In this exercise, you will make acquaintance with the Single-Responsibility Principle and the Open-Closed Principle – the 'S' and 'O' in the SOLID acronym. You will reverse-engineer and refactor a current implementation of a report generator to allow easy implementation of changes to it.

The situation:
You are a SW developer in a company that has developed a report generator tool. This tool can generate reports from a list of employees (name, salary and age), formatted in two different ways – name-first or salary-first. Unfortunately, the previous developer has left the company and has done very little to document the design of the system (and believe me, that is a very common situation!). Being the most recently employed SW developer, and since waste of all sorts roll downhill, you are tasked with implementing some new requirements to the system. You are free to refactor the existing design at will, but you are required to document the changes.

Exercise 1: The Single-Responsibility Principle (SRP)
In this exercise, you will reverse engineer and refactor the existing design to adhere to the SRP

Exercise 1A:
Reverse-engineer the code to a class diagram showing all important classes, their methods and their relations.

Exercise 1B:
Identify all responsibilities of the class `ReportGenerator`.

Exercise 1C:
Refactor the class `ReportGenerator` so that it adheres to the SRP. For the refactored design, state the responsibility of the `ReportGenerator` class and any other class you may have added.

Exercise 1D:
Implement and test your new design.

Exercise 2: The Open-Closed Principle (OCP)

In this exercise, three new requirements are added for the system:

- The system shall hold the age of an employee
- For existing reports (name-first and salary-first) the age shall be last in the report
- It shall be possible to generate a new type of report, age-first, in which the age of an employee is printed first. The order of name and salary after that is not important.

You will implement these new requirements through refactoring your own design from exercise 1 to adhere to the OCP and then extending it. If this is done properly, adding the new requirement should be fairly easy.

Exercise 2A:

Briefly identify any changes necessary to the classes in the *old* design to implement the new requirements.

Exercise 2B:

Refactor your system so that it adheres to OCP and allows a fairly simple implementation of the new requirements.

Exercise 2C:

Implement and test the design from Exercise 2B.