# **Programming Assignment Unit 8**

Computer Science, University of the People

CS 1101-01 Programming Fundamentals - CS 1102-01 - AY2024-T2

Instructor, Noman Shihadeh

January 10, 2023

## Employees - read and filter with streams

In this assignment, we are asked to build a Java application to manage and process employee data. The primary objectives include reading a large dataset of employees, performing specific data manipulations using Java's Function interface and streams, and demonstrating proficiency in these concepts. The program will involve several key operations:

1. Data retrieval – read the data from an external source (csv)
2. Data transform – use different methods to transform and filter the data

**Source Code:**

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.List;

import java.util.function.Function;

import java.util.stream.Stream;

import java.util.Scanner;

public class App {

    private static List<Employee> employees = new ArrayList<>();

    private static Scanner scanner = new Scanner(System.in);

    // Function interface to map an employee to a string (name - department)

    private static Function<Employee, String> nameAndDepartment = employee -> "\* " + employee.getName() + " - "

            + employee.getDepartment() + "\n";

    /\*\*

     \* Main method of the application the main entry point of the application

     \*

     \* @param args

     \*/

    public static void main(String[] args) {

        ReadData("employees.csv");

        // Main loop - keep looping until user enters E

        while (true) {

            System.out.println("Select an option:");

            System.out.println("1 - Employee Details");

            System.out.println("2 - Average Salary");

            System.out.println("3 - Filter Employees");

            System.out.println("E - Exit");

            System.out.println("Enter your choice: ");

            String choice = scanner.next(); // Read user input

            switch (choice.toUpperCase()) {

                case "1":

                    PrintEmployeeDetails();

                    break;

                case "2":

                    CalculateAverageSalary();

                    break;

                case "3":

                    FilterEmployees();

                    break;

                case "E":

                    System.out.println("Bye bye...");

                    return;

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

            hr();

        }

    }

    /\*\*

     \* request user input for minimum age and salary and filter employees based on

     \* the input values and print the filtered employees

     \*/

    private static void FilterEmployees() {

        int minAge = getIntInput("Enter minimum age: ");

        double minSalary = getDoubleInput("Enter minimum salary: ");

        List<Employee> filteredEmployees = employees

                .stream() // Creating a stream from the list of employees

                .filter(employee -> employee.getAge() > minAge) // Filtering employees with age > minAge

                .filter(employee -> employee.getSalary() > minSalary) // Filtering employees with salary > minSalary

                .toList(); // Collecting the stream into a list object

        hr();

        // Printing the filtered employees

        System.out.println("Filtered Employees (Age > " + minAge + ") and (Salary > " + minSalary + "):");

        for (Employee employee : filteredEmployees) {

            System.out.println("\* " + employee.getName() + " - " + employee.getAge());

        }

    }

    /\*\*

     \* Calculate the average salary of all employees and print the result

     \*/

    private static void CalculateAverageSalary() {

        double averageSalary = employees

                .stream() // Creating a stream from the list of employees

                .mapToDouble(Employee::getSalary) // Mapping each employee to a double using the getSalary method

                .average() // Calculating the average of all salaries

                .orElse(0); // If the stream is empty, return 0

        hr();

        // Printing the average salary formatted to 2 decimal places

        System.out.printf("Average Salary: %.2f\n", averageSalary);

    }

    /\*\*

     \* Print the details of all employees in the list

     \*/

    private static void PrintEmployeeDetails() {

        List<String> employeeDetails = employees

                .stream() // Creating a stream from the list of employees

                .map(nameAndDepartment) // Mapping each employee to a string using the function interface

                .toList(); // Collecting the stream into a list object

        hr();

        System.out.println("Employee Details (Name - Department): ");

        for (String employeeDetail : employeeDetails) {

            System.out.print(employeeDetail);

        }

    }

    /\*\*

     \* Read data from a CSV file and populate the employees list

     \*

     \* @param fileName the name and path of the CSV file

     \*/

    private static void ReadData(String fileName) {

        Path path = Paths.get(fileName);

        // Reading CSV file using streams using try-with-resources

        try (Stream<String> lines = Files.lines(path, StandardCharsets.UTF\_8)) {

            employees = lines

                    .skip(1) // Skipping CSV header

                    .map(line -> line.split(",")) // Splitting each line into an array of strings

                    .map(data -> new Employee(data)) // Creating an Employee object from the array of strings

                    .toList(); // Collecting the stream into a list object

        } catch (IOException e) {

            // Log error to console

            System.out.println("Error reading a line in file: " + fileName);

            System.out.println(e.getMessage());

            e.printStackTrace();

            return;

        }

    }

    /\*\*

     \* Print a horizontal line to the console

     \*/

    private static void hr() {

        System.out.println("--------------------------------------------------");

    }

    /\*\*

     \* Get an integer input from the user and validate the input

     \*

     \* @param prompt the prompt to display to the user

     \* @return the integer input from the user

     \*/

    private static int getIntInput(String prompt) {

        while (true) {

            System.out.print(prompt);

            // Check if the input is an integer and if not, display an error message and

            if (!scanner.hasNextInt()) {

                System.out.println("Invalid input. Please enter an integer.");

                scanner.next();

                continue;

            }

            return scanner.nextInt();

        }

    }

    /\*\*

     \* Get a double input from the user and validate the input

     \*

     \* @param prompt the prompt to display to the user

     \* @return the double input from the user

     \*/

    private static double getDoubleInput(String prompt) {

        while (true) {

            System.out.print(prompt);

            // Check if the input is a double and if not, display an error message and

            if (!scanner.hasNextDouble()) {

                System.out.println("Invalid input. Please enter a number.");

                scanner.next();

                continue;

            }

            return scanner.nextDouble();

        }

    }

}

// Employee.java

class Employee {

    private String name;

    private int age;

    private String department;

    private double salary;

    /\*\*

     \* Constructor for Employee class

     \*

     \* @param name       name of the employee

     \* @param age        age of the employee

     \* @param department department of the employee

     \* @param salary     salary of the employee

     \*/

    public Employee(String name, int age, String department, double salary) {

        this.name = name;

        this.age = age;

        this.department = department;

        this.salary = salary;

    }

    /\*\*

     \* Constructor for Employee class that takes an array of strings as input

     \*

     \* @param data array of strings containing the employee data

     \*/

    public Employee(String[] data) {

        this.name = data[0];

        this.age = Integer.parseInt(data[1]);

        this.department = data[2];

        this.salary = Double.parseDouble(data[3]);

    }

    // Getters

    public String getName() {

        return name;

    }

    public int getAge() {

        return age;

    }

    public String getDepartment() {

        return department;

    }

    public double getSalary() {

        return salary;

    }

}

**Output:**

Select an option:

1 - Employee Details

2 - Average Salary

3 - Filter Employees

E - Exit

Enter your choice:

1

--------------------------------------------------

Employee Details (Name - Department):

\* Sarah Jane Smith - Robotics

\* Trinity - Jedi Training

\* Samus Aran - Robotics

\* Rick Deckard - Space Exploration

\* Al Bundy - Paranormal Research

\* Waldo Geraldo Faldo - Time Travel

\* Luke Skywalker - Diplomatic Corps

\* John Sheridan - Robotics

\* Rick Deckard - Planetary Defense

\* Clara Oswald - Time Travel

\* Lara Croft - Robotics

\* Turanga Leela - Planetary Defense

\* Fox Mulder - Android Relations

\* Leia Organa - Diplomatic Corps

\* Sarah Jane Smith - Time Travel

\* Ash Williams - Diplomatic Corps

\* Bart Simpson - Planetary Defense

\* Marty McFly - Paranormal Research

\* Luke Skywalker - Jedi Training

\* Jean-Luc Picard - Planetary Defense

\* Stephanie Tanner - Jedi Training

\* Marty McFly - Time Travel

\* Summer Smith - Jedi Training

\* Pris - Space Exploration

\* Pris - Galactic Relations

\* Clara Oswald - Space Exploration

\* Peggy Bundy - Space Exploration

\* Lisa Simpson - Paranormal Research

\* Fox Mulder - Diplomatic Corps

\* Londo Mollari - Galactic Relations

\* Londo Mollari - Space Exploration

\* Carl Winslow - Starfleet Command

\* Ellen Ripley - Space Exploration

\* Married with Children - Paranormal Research

\* Bender - Android Relations

\* Lisa Simpson - Space Exploration

\* Worf - Paranormal Research

\* Brad Taylor - Space Exploration

\* Data - Jedi Training

\* Laura Winslow - Planetary Defense

\* G'Kar - Galactic Relations

\* Trinity - Jedi Training

\* Trinity - Robotics

\* Brad Taylor - Diplomatic Corps

\* Rick Sanchez - Paranormal Research

\* Sephiroth - Robotics

\* Gordon Freeman - Space Exploration

\* Doc Brown - Starfleet Command

\* Rick Deckard - Starfleet Command

\* Bart Simpson - Space Exploration

\* Luke Skywalker - Space Exploration

\* Neo - Starfleet Command

\* Spock - Jedi Training

\* Luke Skywalker - Planetary Defense

\* Summer Smith - Diplomatic Corps

\* E.T. - Android Relations

\* Roy Batty - Galactic Relations

\* Roy Batty - Starfleet Command

\* Delenn - Android Relations

\* Marge Simpson - Planetary Defense

\* Neytiri - Paranormal Research

\* Homer Simpson - Space Exploration

\* Rose Tyler - Android Relations

\* Neytiri - Time Travel

\* Summer Smith - Robotics

\* Dana Barrett - Time Travel

\* Worf - Space Exploration

\* Eddie Winslow - Android Relations

\* Al Bundy - Planetary Defense

\* Worf - Starfleet Command

\* Eddie Winslow - Diplomatic Corps

\* James T. Kirk - Galactic Relations

\* Delenn - Diplomatic Corps

\* Sonic the Hedgehog - Robotics

\* Rick Sanchez - Space Exploration

\* Cloud Strife - Starfleet Command

\* Turanga Leela - Time Travel

\* Brad Taylor - Time Travel

\* Sonic the Hedgehog - Diplomatic Corps

\* Mark Taylor - Jedi Training

\* D.J. Tanner - Time Travel

\* Ray Stantz - Starfleet Command

\* Randy Taylor - Space Exploration

\* Data - Space Exploration

\* Bud Bundy - Jedi Training

\* Randy Taylor - Planetary Defense

\* Sarah Kerrigan - Robotics

\* Kelly Bundy - Diplomatic Corps

\* Dana Scully - Galactic Relations

\* Bart Simpson - Diplomatic Corps

\* Sephiroth - Android Relations

\* Danny Tanner - Robotics

\* Beth Smith - Galactic Relations

\* Sonic the Hedgehog - Robotics

\* Clara Oswald - Time Travel

\* Rory Williams - Android Relations

\* Bart Simpson - Galactic Relations

\* Homer Simpson - Diplomatic Corps

\* Stephanie Tanner - Planetary Defense

\* Joey Gladstone - Galactic Relations

--------------------------------------------------

Select an option:

1 - Employee Details

2 - Average Salary

3 - Filter Employees

E - Exit

Enter your choice:

2

--------------------------------------------------

Average Salary: 83897.60

--------------------------------------------------

Select an option:

1 - Employee Details

2 - Average Salary

3 - Filter Employees

E - Exit

Enter your choice:

3

Enter minimum age: 33

Enter minimum salary: 3333

--------------------------------------------------

Filtered Employees (Age > 33) and (Salary > 3333.0):

\* Sarah Jane Smith - 42

\* Trinity - 45

\* Samus Aran - 38

\* Rick Deckard - 34

\* Waldo Geraldo Faldo - 39

\* John Sheridan - 35

\* Rick Deckard - 42

\* Lara Croft - 48

\* Fox Mulder - 43

\* Leia Organa - 40

\* Ash Williams - 41

\* Luke Skywalker - 49

\* Jean-Luc Picard - 40

\* Marty McFly - 34

\* Pris - 43

\* Pris - 37

\* Clara Oswald - 43

\* Londo Mollari - 35

\* Londo Mollari - 49

\* Carl Winslow - 34

\* Ellen Ripley - 39

\* Bender - 36

\* Lisa Simpson - 36

\* Worf - 39

\* Brad Taylor - 47

\* Data - 50

\* G'Kar - 37

\* Trinity - 48

\* Rick Sanchez - 34

\* Sephiroth - 43

\* Gordon Freeman - 43

\* Doc Brown - 36

\* Rick Deckard - 48

\* Bart Simpson - 44

\* Luke Skywalker - 40

\* Neo - 47

\* Spock - 38

\* Luke Skywalker - 49

\* Summer Smith - 50

\* E.T. - 44

\* Roy Batty - 48

\* Roy Batty - 41

\* Delenn - 36

\* Marge Simpson - 47

\* Neytiri - 38

\* Rose Tyler - 44

\* Summer Smith - 38

\* Dana Barrett - 42

\* Worf - 49

\* Eddie Winslow - 36

\* Worf - 41

\* James T. Kirk - 45

\* Delenn - 36

\* Rick Sanchez - 48

\* Cloud Strife - 49

\* Mark Taylor - 50

\* D.J. Tanner - 45

\* Randy Taylor - 47

\* Data - 50

\* Bud Bundy - 46

\* Randy Taylor - 35

\* Sarah Kerrigan - 44

\* Dana Scully - 38

\* Bart Simpson - 41

\* Clara Oswald - 48

\* Rory Williams - 46

\* Bart Simpson - 41

\* Homer Simpson - 46

\* Stephanie Tanner - 36

\* Joey Gladstone - 41

--------------------------------------------------

Select an option:

1 - Employee Details

2 - Average Salary

3 - Filter Employees

E - Exit

Enter your choice:

E

Bye bye...

**Code Description:**

The key components of the code are as follows:

1. **Data Storage and Management**:
   * **private static List<Employee> employees**: A list to store the employee objects.
   * **private static Scanner scanner**: A Scanner object for reading user input.
2. **Function<Employee, String> nameAndDepartment**: A function using Java's Function interface to concatenate the employee's name and department.
3. **main(String[] args):** The main entry point and a menu-driven loop that allows the user to choose which of the operations the user wants to execute.
4. **FilterEmployees:** Uses Java streams to filter employees based on age and salary thresholds. Collects and displays the filtered list of employees.
5. **CalculateAverageSalary:** Calculates the average salary of all employees using stream operations. Demonstrates efficient use of Java streams for aggregate operations.
6. **PrintEmployeeDetails:** Uses streams to map each employee to a string containing their name and department. Collects these strings into a list and displays them.
7. **ReadData:** Reads employee data from a CSV file and populates the **employees** list. Demonstrates file reading and data parsing using Java streams.
8. **Helper Methods:**
   * **hr()**: Prints a horizontal rule for better output formatting.
   * **getIntInput(String prompt)** and **getDoubleInput(String prompt)**: Helper methods for input validation and parsing.
9. **Employee class:** the basic structure of the employee whose data is found in the CSV data file.

**Conclusion**

Key parts of the program include the dynamic filtering of employees based on age and salary criteria, the calculation of average salary using stream operations, and the transformation of employee data into a user-friendly format. By using Java's Function interface and streams, the program showcases the ability to handle data transformations, filtering, and aggregate operations effectively. The use of streams not only makes the code more readable and concise but also optimizes performance through lazy evaluation and reduced memory usage.

## References

Java Language and Virtual Machine Specifications

* <https://docs.oracle.com/javase/specs/index.html>

Introduction to Programming Using Java - Version 9.0, JavaFX Edition

* <https://math.hws.edu/javanotes/>

CS 1102-01 - AY2024-T2 - Learning Guide Unit 8

* <https://my.uopeople.edu/mod/book/view.php?id=392003>

**General Disclaimer**: All names and terms that are trademarks are used here for educational and illustrative purposes only, without any intent of infringement. All rights to these trademarks are reserved by their respective owners.

Source Files:

