**Programming Assignment Unit 8**

**Explanation of Function Interface and Streams**

The `Function` interface is a functional interface in Java that represents a function that accepts one argument and produces a result. It is part of the `java.util.function` package and is commonly used in functional programming with Java streams. The `Function` interface is characterized by its single abstract method, `apply`, which takes an input of type `T` and returns a result of type `R`.

Streams in Java, introduced in Java 8, provide a functional approach to processing sequences of elements. They allow for efficient data manipulation and transformation through a series of operations, such as filtering, mapping, and reducing. Streams support parallel execution, making them suitable for handling large datasets.

**Program Requirements**

1. Reading the dataset and storing it in a collection.

2. Writing a function using the Function interface to concatenate an employee's name and department.

3. Generating a new collection with concatenated strings using streams.

4. Finding the average salary of all employees using stream functions.

5. Incorporating a filter function to include employees above a certain age threshold.

**Complete Java Program**

```java

import java.util.\*;

import java.util.function.Function;

import java.util.stream.Collectors;

class Employee {

private String name;

private int age;

private String department;

private double salary;

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

this.name = name;

this.age = age;

this.department = department;

this.salary = salary;

}

public String getName() { return name; }

public int getAge() { return age; }

public String getDepartment() { return department; }

public double getSalary() { return salary; }

}

public class EmployeeProcessing {

public static void main(String[] args) {

// Step 1: Reading the dataset and storing it in a collection

List<Employee> employees = Arrays.asList(

new Employee("Alice", 35, "HR", 60000),

new Employee("Bob", 25, "IT", 50000),

new Employee("Charlie", 32, "Finance", 70000),

new Employee("David", 45, "IT", 80000),

new Employee("Eve", 28, "HR", 55000)

);

// Step 2: Writing a function using the Function interface

Function<Employee, String> nameDeptFunction = emp -> emp.getName() + " (" + emp.getDepartment() + ")";

// Step 3: Generating a new collection with concatenated strings using streams

List<String> nameDeptList = employees.stream()

.map(nameDeptFunction)

.collect(Collectors.toList());

System.out.println("Employee Names and Departments:");

nameDeptList.forEach(System.out::println);

// Step 4: Finding the average salary of all employees using stream functions

double averageSalary = employees.stream()

.mapToDouble(Employee::getSalary)

.average()

.orElse(0.0);

System.out.println("Average Salary: " + averageSalary);

// Step 5: Incorporating a filter function for employees above a certain age threshold

int ageThreshold = 30;

List<Employee> filteredEmployees = employees.stream()

.filter(emp -> emp.getAge() > ageThreshold)

.collect(Collectors.toList());

System.out.println("Employees above age " + ageThreshold + ":");

filteredEmployees.forEach(emp -> System.out.println(emp.getName()));

}

}

```

**Screenshots**

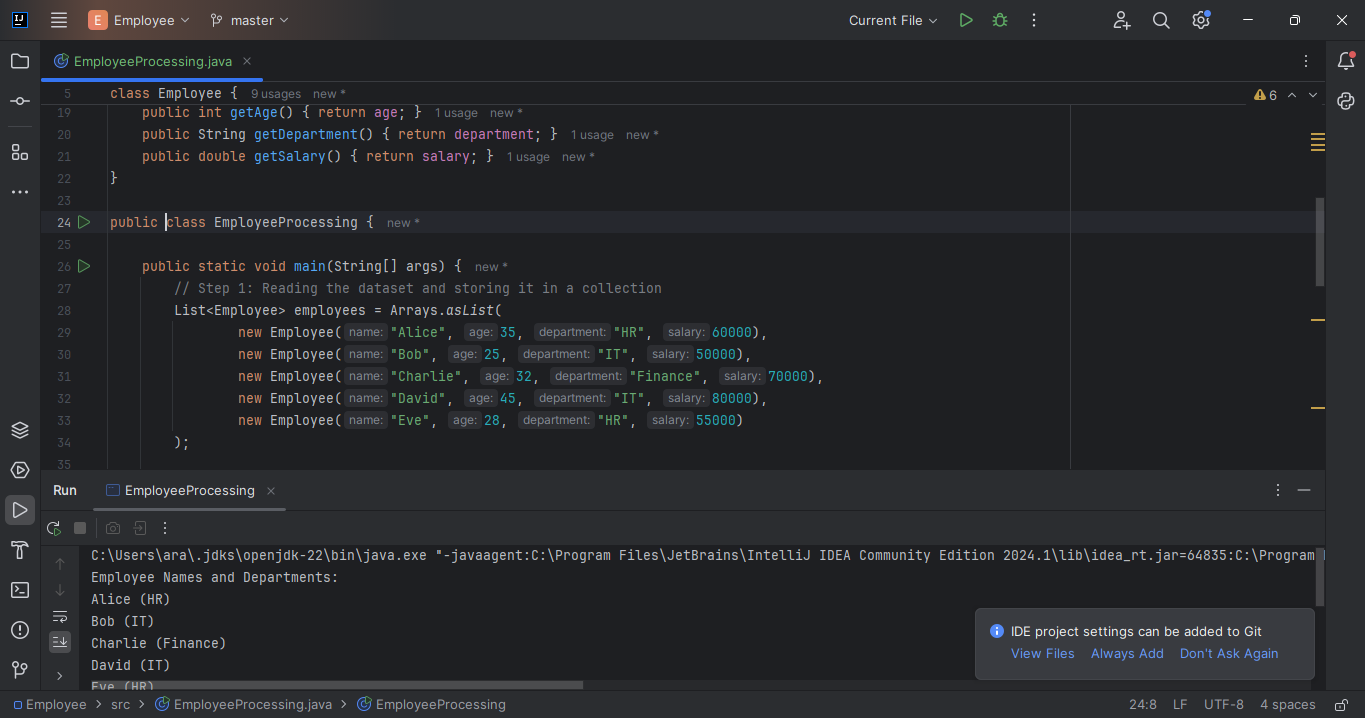
****

Figure 1 - Code Block and Output

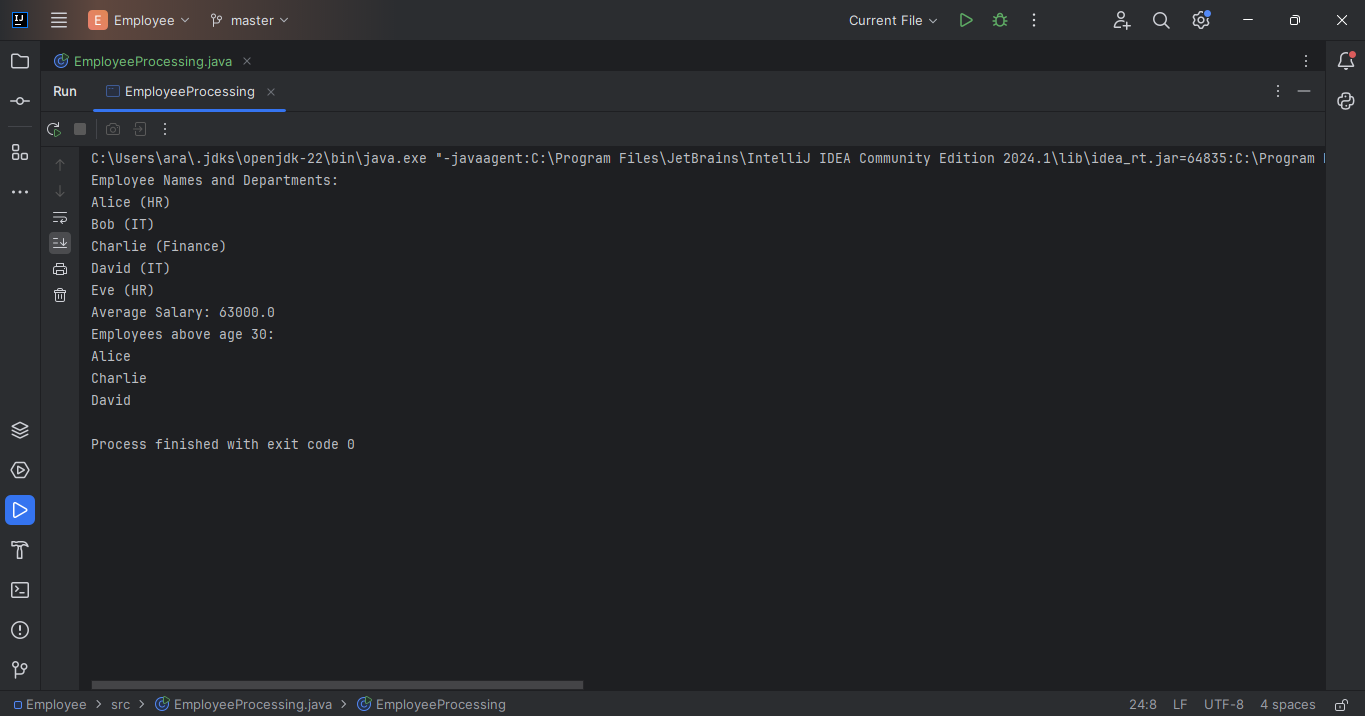
****

Figure 2 - Code Output

**Summary**

The `Function` interface in Java represents a function that takes one argument and returns a result. It is part of the `java.util.function` package and is widely used in conjunction with streams to perform transformations and mappings on collections of data. Streams provide a powerful and expressive way to process sequences of elements, supporting operations like filtering, mapping, and reducing, which can be executed in parallel for enhanced performance.

In the provided program, we first read a dataset of employees and store it in a collection. We then use the `Function` interface to create a function that concatenates an employee's name and department. Using streams, we generate a new collection containing these concatenated strings. We also use stream functions to calculate the average salary of all employees and apply a filter to include only employees above a certain age threshold.

**References:**

Bloch, J. (2018). Effective Java (3rd ed.). Addison-Wesley Professional.

Eck, D. J. (2022). Introduction to Programming Using Java (Version 9, JavaFX Edition). Hobart and William Smith Colleges.