**Programming Assignment Unit 8**

**University of the People**

**CS 1102-01 - AY2024-T5: Programming 1**

**Avernell Christopher- Melikhov**

**August 15th, 2024**

**CODE**

MAIN METHOD

import java.util.List;  
import java.util.stream.Collectors;  
  
public class Main {  
 public static void main(String[] args) {  
  
 // Retrieve the list of employees from the EmployeeData class  
 List<Employee> employees = EmployeeData.*getEmployees*();  
  
 // Create a list of employee names and their departments  
 List<String> nameDepartmentList = employees.stream()  
 .map(EmployeeData.*getNameAndDepartment*)  
 .collect(Collectors.*toList*());  
 System.*out*.println("List of Employees and Department:");  
 nameDepartmentList.forEach(System.*out*::println);  
 System.*out*.println();  
  
 // Create a list of employee names and their job titles  
 List<String> nameJobTitleList = employees.stream()  
 .map(EmployeeData.*getNameAndJobTitle*)  
 .collect(Collectors.*toList*());  
 System.*out*.println("List of Employees and Job Title:");  
 nameJobTitleList.forEach(System.*out*::println);  
 System.*out*.println();  
  
 // Calculate the average salary of all employees  
 double averageSalary = employees.stream()  
 .mapToDouble(Employee::getSalary)  
 .average()  
 .orElse(0.0);  
 System.*out*.println("The average salary is: " + averageSalary);  
 System.*out*.println();  
  
 // Filter employees who are older than 25 years  
 int ageThreshold = 25;  
 List<Employee> filteredEmployeesByAge = employees.stream()  
 .filter(employee -> employee.getAge() > ageThreshold)  
 .collect(Collectors.*toList*());  
 System.*out*.println("Filter by Age:");  
 filteredEmployeesByAge.forEach(employee -> System.*out*.println(employee.getName() + " (" + employee.getAge() + ")"));  
 System.*out*.println();  
  
 // Filter employees who are in the IT department  
 String department = "IT";  
 List<Employee> filteredEmployeesByDepartment = employees.stream()  
 .filter(employee -> employee.getDepartment().equals(department))  
 .collect(Collectors.*toList*());  
 System.*out*.println("Filter by Department:");  
 filteredEmployeesByDepartment.forEach(employee -> System.*out*.println(employee.getName() + " (" + employee.getPosition() + ") - " +  
 "Department: " + employee.getDepartment() + ", " +  
 "Salary: " + employee.getSalary()));  
 }  
}

*EMPLOYEE CLASS*

public class Employee {  
 private String name; // Employee's name  
 private int age; // Employee's age  
 private String department; // Employee's department  
 private double salary; // Employee's salary  
 private String position; // Employee's position  
  
 // Constructor to initialize all fields  
 public Employee(String name, int age, String department, double salary, String position) {  
 this.name = name;  
 this.age = age;  
 this.department = department;  
 this.salary = salary;  
 this.position = position;  
 }  
  
 // Getter for name  
 public String getName() {  
 return name;  
 }  
  
 // Getter for age  
 public int getAge() {  
 return age;  
 }  
  
 // Getter for department  
 public String getDepartment() {  
 return department;  
 }  
  
 // Getter for salary  
 public double getSalary() {  
 return salary;  
 }  
  
 // Getter for position  
 public String getPosition() {  
 return position;  
 }  
  
 // Setter for position  
 public void setPosition(String position) {  
 this.position = position;  
 }  
  
 // Provides a string representation of the employee  
 @Override  
 public String toString() {  
 return name + " (" + position + ")"; // Formats the employee details for display  
 }  
}

*EMPLOYEE DATA CLASS*

import java.util.ArrayList;  
import java.util.List;  
import java.util.function.Function;  
  
public class EmployeeData {  
  
 // Method to create and return a list of Employee objects  
 public static List<Employee> getEmployees() {  
 List<Employee> employees = new ArrayList<>();  
  
 // Adding Employee objects to the list  
 employees.add(new Employee("Jacob", 30, "Finance", 75000, "Chief Financial Officer"));  
 employees.add(new Employee("Alisha", 32, "IT", 64000, "Software Engineer"));  
 employees.add(new Employee("Sam", 42, "HR", 75000, "Chief Human Resource Officer"));  
 employees.add(new Employee("Sasha", 26, "IT", 63000, "IT Specialist"));  
  
 return employees;  
 }  
  
 // Function to get a formatted string with Employee's name and department  
 public static Function<Employee, String> *getNameAndDepartment* = (Employee employee) -> {  
 return employee.getName() + ":" + employee.getDepartment();  
 };  
  
 // Function to get a formatted string with Employee's name and position  
 public static Function<Employee, String> *getNameAndJobTitle* = (Employee employee) -> {  
 return employee.getName() + ":" + employee.getPosition();  
 };  
}