# Experiment 6

**Student Name: Amrit Paudel**                    **UID: 22BCS14341**
**Branch: CSE**                                           **Section: IOT-642 -B**
**Semester: 6<sup>th</sup>**                            **DOP: 10/03/025**
**Subject: Java**                                         **Subject Code:22CSH-359**

## Problem - 6.1

**Aim:** To implement a Java program that sorts a list of Employee objects (based on name, age, and salary) using lambda expressions and stream operations to demonstrate efficient data processing.

## Code:

```java
import java.util.*;
import java.util.stream.Collectors;

class Employee {
    private String name;
    private int age;
    private double salary;

    public Employee(String name, int age, double salary) {
        this.name = name;
        this.age = age;
        this.salary = salary;
    }
    public String getName() { return name; }
    public int getAge() { return age; }
    public double getSalary() { return salary; }

    public void display() {
        System.out.println("Name: " + name + ", Age: " + age + ", Salary: " + salary);
    }
}
```
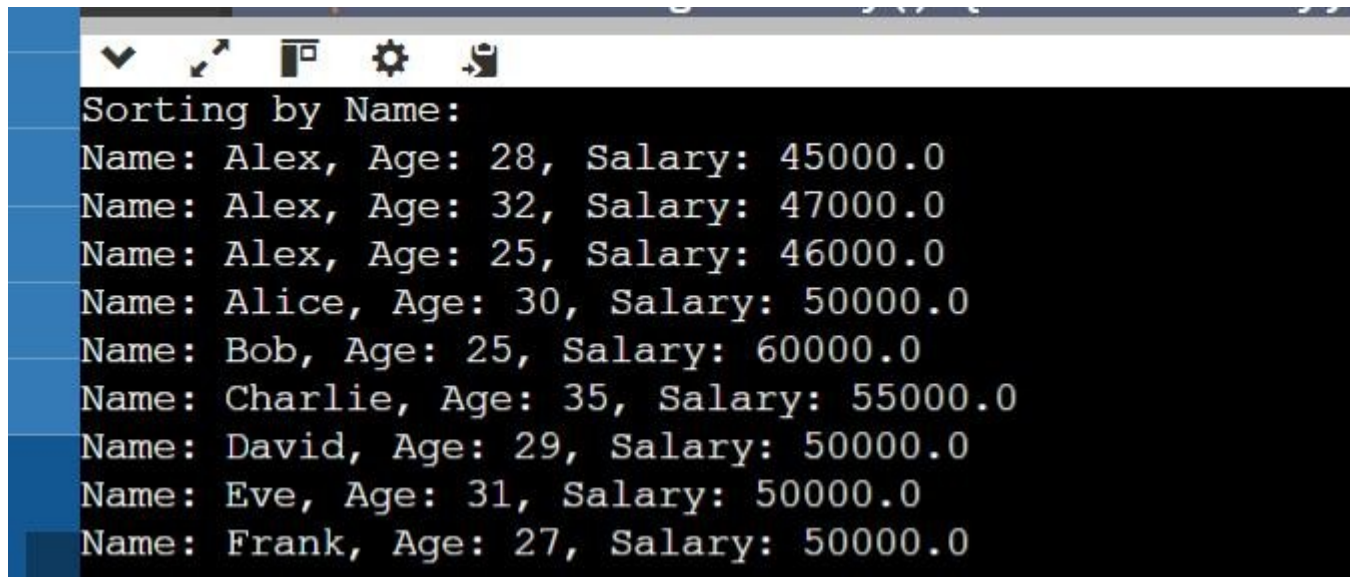
```java
public class EmployeeSorting {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("Alice", 30, 50000));
        employees.add(new Employee("Bob", 25, 60000));
        employees.add(new Employee("Charlie", 35, 55000));
        employees.add(new Employee("Alex", 28, 45000));
        employees.add(new Employee("Alex", 32, 47000));
        employees.add(new Employee("Alex", 25, 46000));
        employees.add(new Employee("David", 29, 50000));
        employees.add(new Employee("Eve", 31, 50000));
        employees.add(new Employee("Frank", 27, 50000));

        System.out.println("Sorting by Name:");
        employees.stream()
            .sorted(Comparator.comparing(Employee::getName))
            .forEach(Employee::display);

        System.out.println("\nSorting by Age:");
        employees.stream()
            .sorted(Comparator.comparingInt(Employee::getAge))
            .forEach(Employee::display);

        System.out.println("\nSorting by Salary:");
        employees.stream()
            .sorted(Comparator.comparingDouble(Employee::getSalary).reversed())
            .forEach(Employee::display);

        System.out.println("\nSorting by Name and then by Age (Edge Case):");
        employees.stream()
            .sorted(Comparator.comparing(Employee::getName).thenComparingInt(Employee::getAge))
            .forEach(Employee::display);

        System.out.println("\nSorting by Salary, then by Name (Edge Case):");
```

```
employees.stream()
    .sorted(Comparator.comparingDouble(Employee::getSalary)
        .thenComparing(Employee::getName))
    .forEach(Employee::display);
    }
}
```

**Output**:

## Problem - 6.2

**Aim** : Implement Java program that uses lambda expressions and Stream API to filter students who scored above 75%, sort them by marks, and display their names.

## Code :

```java
import java.util.*;

import java.util.stream.Collectors;

class Student {

    private String name;

    private double marks

    public Student(String name, double marks) {

        this.name = name;

        this.marks = marks;

    }

    public String getName() { return name; }

    public double getMarks() { return marks;

    public void display() {

        System.out.println("Name: " + name + ", Marks: " + marks);

    }


public class StudentFilterSort {

    public static void main(String[] args) {

        List<Student> students = Arrays.asList(

            new Student("Alice", 80),
```
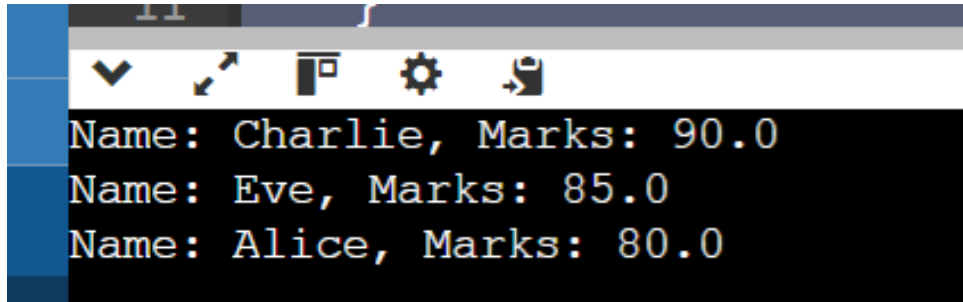
```java
            new Student("Bob", 72),

            new Student("Charlie", 90),

            new Student("David", 65),

            new Student("Eve", 85),

            new Student("Frank", 65)

        );

        List<Student> filteredSortedStudents = students.stream()

            .filter(s -> s.getMarks() > 75)

            .sorted(Comparator.comparingDouble(Student::getMarks).reversed()

                .thenComparing(Student::getName))

            .collect(Collectors.toList());


        if (filteredSortedStudents.isEmpty()) {

            System.out.println("No students scored above 75%.");

        } else {

            filteredSortedStudents.forEach(Student::display);

        }

    }

}
```

**Output :**

```
Name: Charlie, Marks: 90.0
Name: Eve, Marks: 85.0
Name: Alice, Marks: 80.0
```