



Experiment 6

Student Name: Komaldeep Singh
Branch: BE/CSE
Semester: 6th
Subject Name: Project Based
Learning in JAVA with Lab

UID: 22BCS14035
Section/Group: 22BCS_IOT-618-A
Date of Performance: 07/03/25
Subject Code: 22CSH-359

- 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.
- 2. Objective:** The goal of this Java program is to efficiently process and sort a list of Employee objects using modern functional programming techniques, specifically lambda expressions and the Stream API. The program will demonstrate various sorting mechanisms to organize employee data based on different criteria

3. Implementation/Code:

```
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;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        public void display() {
            System.out.println(name + " (" + age + ", " + salary + ")");
        }
    }

    public class EmployeeSorting {
        public static void main(String[] args) {
            List<Employee> employees = new ArrayList<>();

            employees.add(new Employee("Rohan", 35, 55000));
            employees.add(new Employee("Aryan", 28, 45000));
            employees.add(new Employee("Kabir", 32, 47000));
            employees.add(new Employee("Aditya", 25, 46000));
            employees.add(new Employee("Samar", 29, 50000));
            employees.add(new Employee("Vihan", 31, 50000));
            employees.add(new Employee("Ishaan", 27, 50000));

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

            System.out.println();

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

            System.out.println();

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

            System.out.println();

            System.out.println("Sorted by Name then Age:");
            employees.stream()
                .sorted(Comparator.comparing(Employee::getName).thenComparing(Employee::getAge))
                .forEach(Employee::display);

            System.out.println();

            System.out.println("Sorted by Salary then Name:");
            employees.stream()
                .sorted(Comparator.comparingDouble(Employee::getSalary).thenComparing(Employee::getName))
                .forEach(Employee::display);
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}  
}
```

4. Output:

```
Sorted by Name:  
Aditya (25, 46000.0)  
Aryan (28, 45000.0)  
Divit (25, 60000.0)  
Ishaan (27, 50000.0)  
Kabir (32, 47000.0)  
Lakshay (30, 50000.0)  
Rohan (35, 55000.0)  
Samar (29, 50000.0)  
Vihan (31, 50000.0)  
  
Sorted by Age:  
Divit (25, 60000.0)  
Aditya (25, 46000.0)  
Ishaan (27, 50000.0)  
Aryan (28, 45000.0)  
Samar (29, 50000.0)  
Lakshay (30, 50000.0)  
Vihan (31, 50000.0)  
Kabir (32, 47000.0)  
Rohan (35, 55000.0)
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

5.2 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: -

```
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 + " (" + marks + ")");
    }
}

public class StudentFilterSort {
    public static void main(String[] args) {
        List<Student> students = Arrays.asList(
            new Student("Aarav", 80),
            new Student("Vihaan", 72),
            new Student("Rohan", 90),
            new Student("Ishaan", 65),
            new Student("Sanya", 85),
            new Student("Kabir", 70),
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        new Student("Aryan", 78),
        new Student("Samar", 88)
    );

    List<Student> filteredSortedStudents = students.stream()
        .filter(s -> s.getMarks() > 75)
        .sorted(Comparator.comparingDouble(Student::getMarks).revers
ed()
            .thenComparing(Student::getName))
        .collect(Collectors.toList());

    System.out.println("Students scoring above 75%, sorted by
marks:");
    filteredSortedStudents.forEach(Student::display);

}
}
```

Output:

```
Students scoring above 75%, sorted by marks:
Rohan (90.0)
Samar (88.0)
Sanya (85.0)
Aarav (80.0)
Aryan (78.0)
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.