



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 6

Student Name: Divyanshu Jaiswal

Branch: B.E. CSE

Semester: 6th

Subject Name: PBLJ LAB

UID: 22BCS12806

Section/Group: KRG - 2 B

Date of Performance: 10/02/25

Subject Code: 22CSH-359

1. **Aim:** Easy Level: Write a program to sort a list of Employee objects (name, age, salary) using lambda expressions.

2. **Implementation/Code:**

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

class Employee {
    String name;
    int age;
    double salary;
    public Employee(String name, int age, double salary) {
        this.name = name;
        this.salary = salary;
        this.age = age;
    }
    public String toString() {
        return "Name: " + name + ", Age: " + age + ", Salary: " + salary;
    }
}

public class EmployeeSort {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("John", 30, 50000));
        employees.add(new Employee("Alice", 25, 60000));
        employees.add(new Employee("Bob", 35, 45000));

        System.out.println("Sorted by Name:");
        List<Employee> sortedByName = employees.stream()
            .sorted((e1, e2) -> e1.name.compareTo(e2.name))
            .collect(Collectors.toList());

        sortedByName.forEach(System.out::println);
    }
}
```

```
System.out.println("\nSorted by Age:");
List<Employee> sortedByAge = employees.stream()
    .sorted((e1, e2) -> Integer.compare(e1.age, e2.age))
    .collect(Collectors.toList());

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

System.out.println("\nSorted by Salary:");
List<Employee> sortedBySalary = employees.stream()
    .sorted((e1, e2) -> Double.compare(e1.salary, e2.salary))
    .collect(Collectors.toList());

sortedBySalary.forEach(System.out::println);
}
```

3. Output:

```
(#1) { java Employee }
Sorted by Name:
Name: Ravi, Age: 20, Salary: 50000.0
Name: Sahil, Age: 35, Salary: 40000.0
Name: Sumit, Age: 25, Salary: 60000.0

Sorted by Age:
Name: Ravi, Age: 20, Salary: 50000.0
Name: Sumit, Age: 25, Salary: 60000.0
Name: Sahil, Age: 35, Salary: 40000.0

Sorted by Salary:
Name: Sahil, Age: 35, Salary: 40000.0
Name: Ravi, Age: 20, Salary: 50000.0
Name: Sumit, Age: 25, Salary: 60000.0
```

3. **Aim:** Create a program to use lambda expressions and stream operations to filter students scoring above 75%, sort them by marks, and display their names.

4. Implementation/Code:

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

class Student {
    String name;
    double percentage;
```

```
public Student(String name, double percentage) {
    this.name = name;
    this.percentage = percentage;
}

public String toString() {
    return "Name: " + name + ", Percentage: " + percentage;
}
}

public class StudentFilterSort {
    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();

        students.add(new Student("Shreya", 92.5));
        students.add(new Student("Aditi", 85.0));
        students.add(new Student("Ansh", 90.0));
        students.add(new Student("Raju", 78.5));
        System.out.println("Students scoring above 75%,sorted by marks:");
        students.stream()
            .filter(student -> student.percentage > 75)
            .sorted((s1, s2) -> Double.compare(s2.percentage, s1.percentage))
            .map(student -> student.name)
            .forEach(System.out::println);
    }
}
```

5. Output:

```
Students scoring above 75%, sorted by marks:
Ravi
Anita
Aashu
kunal
```