**EXPERIMENT- 8**

**Student Name:** Ayush guleria **UID:** 22BCS16080

**Branch:** BE CSE **Section/Group:** 22BCS\_IOT-637-A

**Semester:** 6TH **Date of Performance:** 10.03.25

**Subject Name:** Project Based Learning in Java **Subject Code:**22CSH-359

**EASY LEVEL**

1. **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.

**Objective:** To develop a **Java program** that utilizes **lambda expressions** and **stream operations** to:

1. **Filter** students who scored **above 75%**.
2. **Sort** the filtered students based on their marks **in ascending order**.
3. **Display** only the **names** of the selected students.

# Implementation/Code:

import java.util.\*;

import java.util.stream.\*;

class Student {

String name;

double marks;

Student(String name, double marks) {

this.name = name;

this.marks = marks;

}

public String getName() {

return name;

}

public double getMarks() {

return marks;

}

}

public class StudentFilter {

public static void main(String[] args) {

List<Student> students = Arrays.asList(

new Student("Alice", 82.5),

new Student("Bob", 67.0),

new Student("Charlie", 91.0),

new Student("David", 74.9),

new Student("Eva", 77.3)

);

System.out.println("Students scoring above 75% (sorted by marks):");

students.stream()

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

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

.forEach(s -> System.out.println(s.getName() + " - " + s.getMarks() + "%"));

}

}

# Output:

1. **Learning Outcomes:**
   * By implementing this program, you will learn:
   * **Lambda Expressions:** Understand how to use concise, functional-style coding in Java.
   * **Streams API:** Learn to process collections efficiently using **filter(), sorted(), and forEach()**.
   * **Filtering Data:** Apply the **filter()** method to select specific elements from a list.
   * **Sorting with Comparator:** Use **sorted()** with method references to arrange data.
   * **Functional Programming Approach:** Gain hands-on experience in writing **clean and efficient** Java code.