



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment-6

Student Name: Prateek Pratap Singh

Branch: BE-CSE

Semester: 6th

**Subject Name: Project Based Learning
in Java with Lab**

UID: 22BCS10036

Section/Group: IOT_631-A

Date of Performance: 21/03/2025

Subject Code: 22CSH-359

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

2. Implementation:

```
import java.util.*;

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

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

    public String toString() {
        return name + " " + age + " " + salary;
    }
}

public class Main {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("Prateek", 21, 50000));
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
employees.add(new Employee("Yash", 25, 60000));  
employees.add(new Employee("Rohan", 35, 55000));  
  
employees.sort(Comparator.comparingDouble(e -> e.salary));  
employees.forEach(System.out::println);  
}  
}
```

3. Output:

```
Prateek 21 50000.0  
Rohan 35 55000.0  
Yash 25 60000.0  
  
** Process exited - Return Code: 0 **
```

1. (b) **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

2. Implementation :

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

class Student {
    String name;
    double marks;

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

public class Main {
    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();
        students.add(new Student("Prateek", 80));
        students.add(new Student("Rohan", 65));
        students.add(new Student("Yash", 90));
        students.add(new Student("Tej", 70));

        students.stream()
            .filter(s -> s.marks > 75)
            .sorted(Comparator.comparingDouble(s -> -s.marks))
            .map(s -> s.name)
            .forEach(System.out::println);
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

3. Output :

A screenshot of a code execution environment. At the top, there is a green 'Run' button with a play icon, a grey 'Share' button with a share icon, and a text input field labeled 'Command Line Arguments'. Below these buttons is a dark-themed output window. On the left side of this window is a vertical toolbar with icons for file operations (copy, download, upload, share). The output text is as follows:

```
Prateek  
Yash  
  
** Process exited - Return Code: 0 **
```