



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment- 06

Student Name: Pargat Singh

UID: 22ICS10002

Branch: BE-CSE

Section/Group: IOT-641/B

Semester: 6<sup>th</sup>

Date of Performance: 06/03/2025

Subject Name: Project Based Learning in JAVA

Code: 22CSH-359

with Lab.

1. **Aim(EASY LEVEL)** :Write a program to sort a list of Employee objects (name, age, salary) using lambda expressions.
2. **Objective:** To efficiently sort a list of Employee objects based on different attributes (name, age, salary) using concise and readable lambda expressions in Java.
3. **Implementation/Code:**

```
import java.util.ArrayList;  
import java.util.Scanner;
```

```
class Employee {  
    int id;  
    String name;  
    double salary;
```

```
    Employee(int id, String name, double salary) {  
        this.id = id;  
        this.name = name;  
        this.salary = salary;  
    }
```

```
    public String toString() {  
        return "ID: " + id + ", Name: " + name + ", Salary: " + salary;  
    }  
}
```

```
public class EmployeeManager {  
    private static ArrayList<Employee> employees = new ArrayList<>();  
    private static Scanner scanner = new Scanner(System.in);
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
public static void main(String[] args) {
    while (true) {
        System.out.println("\n1. Add Employee");
        System.out.println("2. Update Employee");
        System.out.println("3. Remove Employee");
        System.out.println("4. Search Employee");
        System.out.println("5. Display All Employees");
        System.out.println("6. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        switch (choice) {
            case 1: addEmployee(); break;
            case 2: updateEmployee(); break;
            case 3: removeEmployee(); break;
            case 4: searchEmployee(); break;
            case 5: displayEmployees(); break;
            case 6: System.out.println("Exiting..."); return;
            default: System.out.println("Invalid choice! Try again.");
        }
    }
}

private static void addEmployee() {
    System.out.print("Enter ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Enter Name: ");
    String name = scanner.nextLine();
    System.out.print("Enter Salary: ");
    double salary = scanner.nextDouble();
    employees.add(new Employee(id, name, salary));
    System.out.println("Employee added successfully!");
}

private static void updateEmployee() {
    System.out.print("Enter Employee ID to update: ");
    int id = scanner.nextInt();
    for (Employee emp : employees) {
        if (emp.id == id) {
            scanner.nextLine();
            System.out.print("Enter New Name: ");
            emp.name = scanner.nextLine();
            System.out.print("Enter New Salary: ");
            emp.salary = scanner.nextDouble();
            System.out.println("Employee updated successfully!");
        }
    }
}
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        return;
    }
}
System.out.println("Employee not found!");
}

private static void removeEmployee() {
    System.out.print("Enter Employee ID to remove: ");
    int id = scanner.nextInt();
    employees.removeIf(emp -> emp.id == id);
    System.out.println("Employee removed successfully!");
}

private static void searchEmployee() {
    System.out.print("Enter Employee ID to search: ");
    int id = scanner.nextInt();
    for (Employee emp : employees) {
        if (emp.id == id) {
            System.out.println(emp);
            return;
        }
    }
    System.out.println("Employee not found!");
}

private static void displayEmployees() {
    if (employees.isEmpty()) {
        System.out.println("No employees found.");
    } else {
        for (Employee emp : employees) {
            System.out.println(emp);
        }
    }
}
}
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## 4. Output:

```
(base) PS D:\React project> cd "d:\React project\java\java4\java6\" ; if ($?) { javac EmployeeSorter.java } ; if ($?) { java EmployeeSorter }
Sorted by Name:
Deepanjali (22, 45000.0)
Deepu (30, 70000.0)
Naman (35, 80000.0)
Pargat (25, 50000.0)
(base) PS D:\React project\java\java4\java6>
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

**AIM(MEDIUM LEVEL)-** Create a program to use lambda expressions and stream operations to filter students scoring above 75%, sort them by marks, and display their names.

## Implementation/Code:

```
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 StudentFilter {
    public static void main(String[] args) {
        List<Student> students = Arrays.asList(
            new Student("Pargat", 85),
            new Student("Deepu", 74),
            new Student("Deepanjali", 90),
            new Student("Naman", 78)
        );

        // Filtering students with marks above 75%, sorting by marks, and displaying names
        students.stream()
            .filter(s -> s.marks > 75)
            .sorted(Comparator.comparingDouble(s -> -s.marks))
            .map(s -> s.name)
            .forEach(System.out::println);
    }
}
```

## Output:

```
cd "d:
\React project\java\java4\java6\" ; if ($?) { javac
StudentFilter.java } ; if ($?) { java StudentFilde
r }
Deepanjali
Pargat
Naman
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

**Aim(HARD LEVEL):** Write a Java program to process a large dataset of products using streams. Perform operations such as grouping products by category, finding the most expensive product in each category, and calculating the average price of all products..

## Implementation/Code:

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

class Product {
    String name;
    String category;
    double price;

    public Product(String name, String category, double price) {
        this.name = name;
        this.category = category;
        this.price = price;
    }

    @Override
    public String toString() {
        return name + " (" + category + ", $" + price + ")";
    }
}

public class ProductProcessor {
    public static void main(String[] args) {
        List<Product> products = Arrays.asList(
            new Product("Laptop", "Electronics", 1200),
            new Product("Phone", "Electronics", 800),
            new Product("TV", "Electronics", 1500),
            new Product("Shirt", "Clothing", 50),
            new Product("Jeans", "Clothing", 80),
            new Product("Blender", "Home Appliances", 100),
            new Product("Vacuum Cleaner", "Home Appliances", 200)
        );

        // Grouping products by category
        Map<String, List<Product>> productsByCategory = products.stream()
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
.collect(Collectors.groupingBy(p -> p.category));

System.out.println("Products grouped by category:");
productsByCategory.forEach((category, list) -> {
    System.out.println(category + ": " + list);
});

// Finding the most expensive product in each category
Map<String, Optional<Product>> mostExpensiveByCategory = products.stream()
    .collect(Collectors.groupingBy(
        p -> p.category,
        Collectors.maxBy(Comparator.comparingDouble(p -> p.price))
    ));

System.out.println("\nMost expensive product in each category:");
mostExpensiveByCategory.forEach((category, product) ->
    System.out.println(category + ": " + product.orElse(null)));

// Calculating the average price of all products
double averagePrice = products.stream()
    .mapToDouble(p -> p.price)
    .average()
    .orElse(0);

System.out.println("\nAverage price of all products: $" + averagePrice);
}
}
```

## Output:

```
(base) PS D:\React project> cd "d:\React project\java\java4\java6\" ; if ($?) { javac ProductProcessor.java } ; if ($?) { java ProductProcessor }
Products grouped by category:
Clothing: [Shirt (Clothing, $50.0), Jeans (Clothing, $80.0)]
Electronics: [Laptop (Electronics, $1200.0), Phone (Electronics, $800.0), TV (Electronics, $1500.0)]
Home Appliances: [Blender (Home Appliances, $100.0), Vacuum Cleaner (Home Appliances, $200.0)]

Most expensive product in each category:
Clothing: Jeans (Clothing, $80.0)
Electronics: TV (Electronics, $1500.0)
Home Appliances: Vacuum Cleaner (Home Appliances, $200.0)

Average price of all products: $561.4285714285714
```



### 5. Learning Outcomes:

1. Mastering Lambda Expressions – Learn how to use lambda functions for sorting and filtering data efficiently.
2. Working with Java Streams API – Understand how to process collections using stream operations like filtering, sorting, and mapping.
3. Grouping and Aggregation – Use `Collectors.groupingBy()` to categorize data and `Collectors.maxBy()` to find the highest value in a group.
4. Functional Programming Concepts – Apply functional programming techniques like method references and `mapToDouble()` for calculations.
5. Handling Optional Values – Use `Optional` to avoid `NullPointerException` while finding max values in each category.





DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.