NAME : Dhreeti Garg	**UID :** 22BCS16521	**SECTION :** 607-A

Easy Level: Sort a list of Employee objects based on different attributes

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

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

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

    public String toString() {
        return name + ", Age: " + age + ", Salary: $" + salary;
    }
}

public class EmployeeSort {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("Alice", 30, 70000));
        employees.add(new Employee("Bob", 25, 50000));
        employees.add(new Employee("Charlie", 35, 80000));
        employees.add(new Employee("David", 28, 60000));

        // Sort by Name
        Collections.sort(employees, (e1, e2) -> e1.name.compareTo(e2.name));
        System.out.println("Sorted by Name:");
        employees.forEach(System.out::println);

        // Sort by Age
        Collections.sort(employees, (e1, e2) -> Integer.compare(e1.age, e2.age));
        System.out.println("\nSorted by Age:");
        employees.forEach(System.out::println);

        // Sort by Salary
        Collections.sort(employees, (e1, e2) -> Double.compare(e1.salary, e2.salary));
        System.out.println("\nSorted by Salary:");
        employees.forEach(System.out::println);
    }
}
```

```
Sorted by Name:
Alice, Age: 30, Salary: $70000.0
Bob, Age: 25, Salary: $50000.0
Charlie, Age: 35, Salary: $80000.0
David, Age: 28, Salary: $60000.0

Sorted by Age:
Bob, Age: 25, Salary: $50000.0
David, Age: 28, Salary: $60000.0
Alice, Age: 30, Salary: $70000.0
Charlie, Age: 35, Salary: $80000.0

Sorted by Salary:
Bob, Age: 25, Salary: $50000.0
David, Age: 28, Salary: $60000.0
Alice, Age: 30, Salary: $70000.0
Charlie, Age: 35, Salary: $80000.0
PS E:\Web development\java>
```

Medium Level: Filters students scoring above 75%, sorts them by mark

```
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;

class Student {
    String name;
    double marks;

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

public class StudentFilterAndSort {
    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();
        students.add(new Student("Alice", 88.5));
        students.add(new Student("Bob", 72.0));
        students.add(new Student("Charlie", 91.0));
        students.add(new Student("David", 68.5));
        students.add(new Student("Eva", 79.0));

        List<String> topStudents = students.stream()
```

```
                .filter(s -> s.marks > 75)
                .sorted(Comparator.comparingDouble(s -> s.marks))
                .map(s -> s.name)
                .collect(Collectors.toList());

        System.out.println("Students scoring above 75% (Sorted by Marks):");
        topStudents.forEach(System.out::println);
    }
}
```

```
Students scoring above 75% (Sorted by Marks):
Eva
Alice
Charlie
PS E:\Web development\java>
```

Hard Level: Process a large dataset of products using streams

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

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

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

    public String toString() {
        return name + " ($" + price + ")";
    }
}

public class ProductStreamOperations {
    public static void main(String[] args) {
        List<Product> products = Arrays.asList(
            new Product("Laptop", "Electronics", 1500),
            new Product("Phone", "Electronics", 800),
            new Product("Desk", "Furniture", 300),
            new Product("Chair", "Furniture", 150),
```

```java
        new Product("Shirt", "Clothing", 50),
        new Product("Jacket", "Clothing", 120)
    );

    // Grouping by Category
    Map<String, List<Product>> groupedByCategory = products.stream()
        .collect(Collectors.groupingBy(p -> p.category));

    System.out.println("Grouped by Category:");
    groupedByCategory.forEach((category, productList) -> {
        System.out.println(category + " -> " + productList);
    });

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

    System.out.println("\nMost Expensive Product in Each Category:");
    mostExpensive.forEach((category, product) ->
        System.out.println(category + " -> " + product.orElse(null))
    );

    // Calculating average price of all products
    double averagePrice = products.stream()
        .collect(Collectors.averagingDouble(p -> p.price));

    System.out.println("\nAverage Price of All Products: $" + averagePrice);
  }
}
```

```
Grouped by Category:
Clothing -> [Shirt ($50.0), Jacket ($120.0)]
Electronics -> [Laptop ($1500.0), Phone ($800.0)]
Furniture -> [Desk ($300.0), Chair ($150.0)]

Most Expensive Product in Each Category:
Clothing -> Jacket ($120.0)
Electronics -> Laptop ($1500.0)
Furniture -> Desk ($300.0)

Average Price of All Products: $486.6666666666667
PS E:\Web development\java>
```