Q1. Sort a list of students by roll number (ascending) using Comparable.

Create a Student class with fields: rollNo, name, and marks. Implement the Comparable interface to sort students by their roll numbers.

import java.util.\*;

class Student implements Comparable<Student>

{

int id;

String name;

Student(int id, String name)

{

this.id = id;

this.name = name;

}

public int compareTo(Student s)

{

return this.name.compareTo(s.name);

}

public String toString()

{

return id + " " + name;

}

public static void main (String[] args)

{

List<Student> list = Arrays.*asList*(

new Student(6,"ajay"),

new Student(2,"arun"),

new Student(7,"arjun")

);

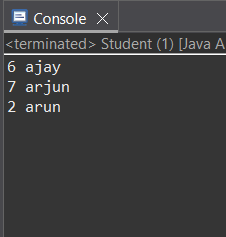
Collections.*sort*(list);

list.forEach(System.***out***::println);

}

}

**OUTPUT:**



Q2. Create a Product class and sort products by price using Comparable.

Implement Comparable<Product> and sort a list of products using Collections.sort().

import java.util.\*;

class Product implements Comparable<Product> {

    private int id;

    private String name;

    private double price;

    public Product(int id, String name, double price) {

        this.id = id;

        this.name = name;

        this.price = price;

    }

    public double getPrice() {

        return price;

    }

    public String getName() {

        return name;

    }

    @Override

    public int compareTo(Product other) {

        return Double.compare(this.price, other.price); // Ascending order

    }

    @Override

    public String toString() {

        return "Product{id=" + id + ", name='" + name + "', price=" + price + '}';

    }

}

public class Comparable\_product {

    public static void main(String[] args) {

        List<Product> products = new ArrayList<>();

        products.add(new Product(101, "Laptop", 50000));

        products.add(new Product(102, "Smartphone", 60000));

        products.add(new Product(103, "Tablet", 35000));

        products.add(new Product(104, "Smartwatch", 25000));

        Collections.sort(products);

        System.out.println("Products sorted by price :");

        for (Product p : products) {

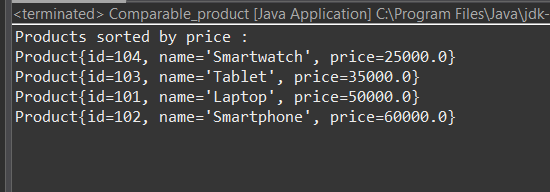
            System.out.println(p);

        }

    }

}

**OUTPUT:**



Q3. Create an Employee class and sort by name using Comparable.

Use the compareTo() method to sort alphabetically by employee names.

import java.util.\*;

class Employee implements Comparable<Employee> {

private int id;

private String name;

private double salary;

public Employee(int id, String name, double salary) {

this.id = id;

this.name = name;

this.salary = salary;

}

public String getName() {

return name;

}

public int compareTo(Employee other) {

return this.name.compareToIgnoreCase(other.name); // Case-insensitive alphabetical order

}

public String toString() {

return "Employee{id=" + id + ", name='" + name + "', salary=" + salary + "}";

}

}

public class Comparable\_Employee {

public static void main(String[] args) {

List<Employee> emps = new ArrayList<>();

emps.add(new Employee(101, "Abc", 70000));

emps.add(new Employee(102, "def", 65000));

emps.add(new Employee(103, "hij", 60000));

emps.add(new Employee(104, "klm", 75000));

Collections.sort(emps);

System.out.println("Employees sorted by name :");

for (Employee emp : emps) {

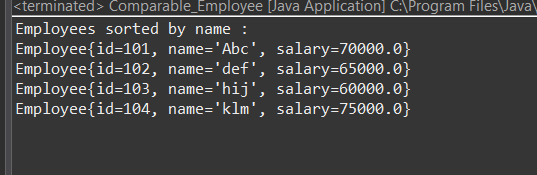
System.out.println(emp);

}

}

}

**OUTPUT:**



Q4. Sort a list of Book objects by bookId in descending order using Comparable.

Hint: Override compareTo() to return the reverse order.

import java.util.\*;

class Book implements Comparable<Book> {

private int bookId;

private String title;

public Book(int bookId, String title) {

this.bookId = bookId;

this.title = title;

}

public int getBookId() {

return bookId;

}

public int compareTo(Book other) {

return Integer.compare(other.bookId, this.bookId);

}

public String toString() {

return "Book{id=" + bookId + ", title='" + title + "'}";

}

}

public class BookIDdemo {

public static void main(String[] args) {

List<Book> books = new ArrayList<>();

books.add(new Book(103, "book1"));

books.add(new Book(101, "book2"));

books.add(new Book(104, "book3"));

Collections.sort(books);

System.out.println("Books sorted by bookId in Desc order :");

for (Book book : books) {

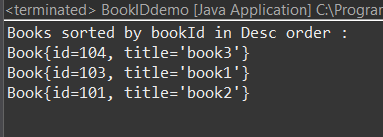
System.out.println(book);

}

}

}

**OUTPUT:**



Q5. Implement a program that sorts a list of custom objects using Comparable, and displays them before and after sorting.

import java.util.\*;

class Employees implements Comparable<Employees> {

private int empId;

private String empName;

private int empSalary;

public Employees(int empId, String empName, int empSalary) {

this.empId = empId;

this.empName = empName;

this.empSalary = empSalary;

}

@Override

public int compareTo(Employees other) {

return Integer.compare(this.empSalary, other.empSalary);

}

@Override

public String toString() {

return "Employees{id=" + empId + ", name='" + empName + "', salary=" + empSalary + "}";

}

}

public class ComparableCustomObject {

public static void main(String[] args) {

List<Employees> employeeList = new ArrayList<Employees>();

employeeList.add(new Employees(201, "Ravi", 75000));

employeeList.add(new Employees(202, "Sneha", 88000));

employeeList.add(new Employees(203, "Kiran", 67000));

employeeList.add(new Employees(204, "Meera", 92000));

System.out.println("Before Sorting:");

for (Employees emp : employeeList) {

System.out.println(emp);

}

Collections.sort(employeeList);

System.out.println("\nAfter Sorting by Salary:");

for (Employees emp : employeeList) {

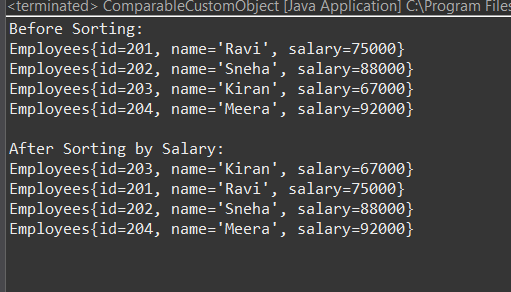
System.out.println(emp);

}

}

}

**OUTPUT:**



Q6. Sort a list of students by marks (descending) using Comparator.

Create a Comparator class or use a lambda expression to sort by marks.

import java.util.\*;

class Student1 {

int id;

String name;

Student1(int id, String name) {

this.id = id;

this.name = name;

}

public String toString() {

return id + " " + name;

}

}

public class TestComparator implements Comparator<Student1> {

public int compare(Student1 s1,Student1 s2)

{

return Integer.compare(s1.id,s2.id);

}

public static void main(String[] args) {

List<Student1> list =new ArrayList<>();

list.add(new Student1(0, "arjun"));

list.add(new Student1(3,"vijay"));

list.add(new Student1(1, "manas"));

list.add(new Student1(2, "Kittu"));

System.out.println(list);

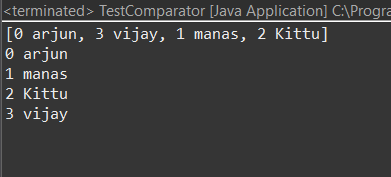
Collections.sort(list, new TestComparator());

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

}

}

**OUTPUT:**



Create multiple sorting strategies for a Product class.

Implement comparators to sort by:

Price ascending

Price descending

Name alphabetically

import java.util.\*;

class Item {

private int id;

private String name;

private double price;

public Item(int id, String name, double price) {

this.id = id;

this.name = name;

this.price = price;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public double getPrice() {

return price;

}

public String toString() {

return "Product{id=" + id + ", name='" + name + "', price=" + price + "}";

}

public static final Comparator<Item> PRICE\_ASCENDING = new Comparator<Item>() {

public int compare(Item p1, Item p2) {

return Double.compare(p1.getPrice(), p2.getPrice());

}

};

public static final Comparator<Item> PRICE\_DESCENDING = new Comparator<Item>() {

public int compare(Item p1, Item p2) {

return Double.compare(p2.getPrice(), p1.getPrice());

}

};

public static final Comparator<Item> NAME\_ALPHABETICALLY = new Comparator<Item>() {

public int compare(Item p1, Item p2) {

return p1.getName().compareToIgnoreCase(p2.getName());

}

};

}

public class comparator7 {

public static void main(String[] args) {

List<Item> products = new ArrayList<>();

products.add(new Item(101, "laptop", 50000));

products.add(new Item(102, "phone", 60000));

products.add(new Item(103, "tablet", 35000));

products.add(new Item(104, "watch", 25000));

System.out.println("Original list:");

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

Collections.sort(products, Item.PRICE\_ASCENDING);

System.out.println("\nSorted by Price (Asc):");

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

Collections.sort(products, Item.PRICE\_DESCENDING);

System.out.println("\nSorted by Price (Desc):");

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

Collections.sort(products, Item.NAME\_ALPHABETICALLY);

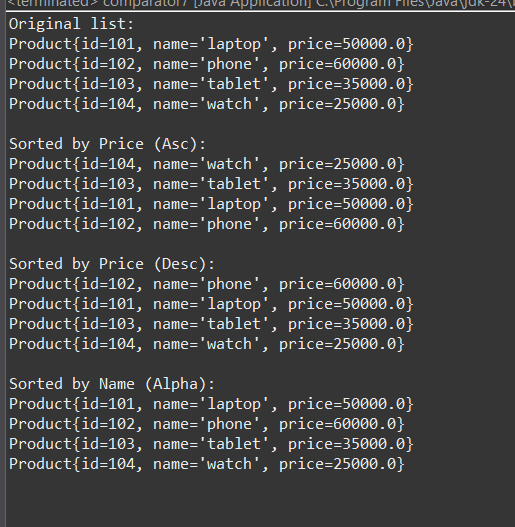
System.out.println("\nSorted by Name (Alpha):");

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

}

}

**OUTPUT:**



Sort Employee objects by joining date using Comparator.

Use Comparator to sort employees based on LocalDate or Date.

import java.time.LocalDate;

import java.util.\*;

class Employe {

private int id;

private String name;

private LocalDate joiningDate;

public Employe(int id, String name, LocalDate joiningDate) {

this.id = id;

this.name = name;

this.joiningDate = joiningDate;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public LocalDate getJoiningDate() {

return joiningDate;

}

public String toString() {

return "Employee{id=" + id + ", name='" + name + "', joiningDate=" + joiningDate + "}";

}

}

public class Com\_Emp {

public static void main(String[] args) {

List<Employe> emp = new ArrayList<>();

emp.add(new Employe(101, "vijju", LocalDate.of(2025, 5, 10)));

emp.add(new Employe(102, "harish", LocalDate.of(2025, 7, 15)));

emp.add(new Employe(103, "nisha", LocalDate.of(2025, 9, 1)));

emp.add(new Employe(104, "raj", LocalDate.of(2025, 1, 20)));

System.out.println("Before Sorting by Joining Date:");

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

emp.sort(Comparator.comparing(Employe::getJoiningDate));

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

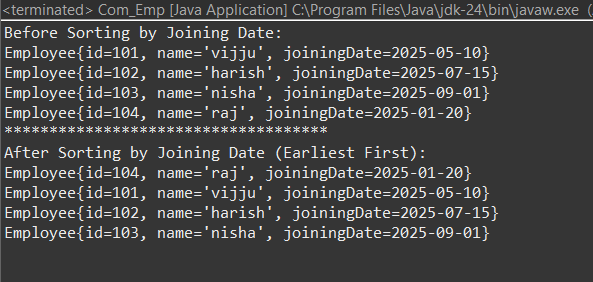
System.out.println("After Sorting by Joining Date (Earliest First):");

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

}

}

**OUTPUT:**



Write a program that sorts a list of cities by population using Comparator.

import java.util.\*;

class City {

private String name;

private int population;

public City(String name, int population) {

this.name = name;

this.population = population;

}

public String getName() {

return name;

}

public int getPopulation() {

return population;

}

public String toString() {

return "City{name='" + name + "', population=" + population + "}";

}

}

public class Cities {

public static void main(String[] args) {

List<City> c = new ArrayList<>();

c.add(new City("Tokyo", 37400068));

c.add(new City("New York", 18900000));

c.add(new City("London", 9300000));

c.add(new City("Paris", 11000000));

c.add(new City("Sydney", 5400000));

c.add(new City("Dubai", 3500000));

c.add(new City("Singapore", 5900000));

System.out.println("Before Sorting by Population:");

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

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

c.sort(Comparator.comparing(City::getPopulation).reversed());

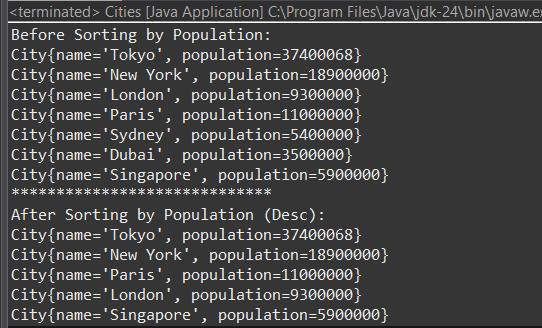
System.out.println("After Sorting by Population (Desc):");

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

}

}

**OUTPUT:**



Create a program where:

Student implements Comparable to sort by name

Use Comparator to sort by marks

Demonstrate both sorting techniques in the same program.

import java.util.\*;

class Studentss implements Comparable<Studentss> {

private int rollNo;

private String name;

private int marks;

public Studentss(int rollNo, String name, int marks) {

this.rollNo = rollNo;

this.name = name;

this.marks = marks;

}

public String getName() {

return name;

}

public int getMarks() {

return marks;

}

public String toString() {

return "Student{rollNo=" + rollNo + ", name='" + name + "', marks=" + marks + "}";

}

public int compareTo(Studentss other) {

return this.name.compareToIgnoreCase(other.name);

}

public static final Comparator<Studentss> MARKS\_COMPARATOR = new Comparator<Studentss>() {

public int compare(Studentss s1, Studentss s2) {

return Integer.compare(s1.getMarks(), s2.getMarks());

}

};

}

public class Student\_Demo {

public static void main(String[] args) {

List<Studentss> s = new ArrayList<>();

s.add(new Studentss(101, "Ravi", 85));

s.add(new Studentss(102, "Anita", 92));

s.add(new Studentss(103, "Mohit", 78));

s.add(new Studentss(104, "Geeta", 90));

System.out.println("Original List:");

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

Collections.sort(s);

System.out.println("\nSorted by Name (Alpha):");

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

Collections.sort(s, Studentss.MARKS\_COMPARATOR);

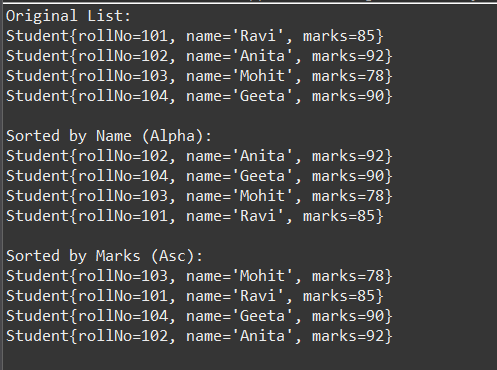
System.out.println("\nSorted by Marks (Asc):");

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

}

}

**OUTPUT:**



Sort a list of Book objects using both Comparable (by ID) and Comparator (by title, then author).

import java.util.\*;

class Books implements Comparable<Books> {

private int bookId;

private String title;

private String author;

public Books(int bookId, String title, String author) {

this.bookId = bookId;

this.title = title;

this.author = author;

}

public int getBookId() {

return bookId;

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

public int compareTo(Books other) {

return Integer.compare(this.bookId, other.bookId);

}

public String toString() {

return "Book{id=" + bookId + ", title='" + title + "', author='" + author + "'}";

}

public static final Comparator<Books> TITLE\_THEN\_AUTHOR\_COMPARATOR = new Comparator<Books>() {

public int compare(Books b1, Books b2) {

int titleComparison = b1.getTitle().compareToIgnoreCase(b2.getTitle());

if (titleComparison != 0) {

return titleComparison;

}

return b1.getAuthor().compareToIgnoreCase(b2.getAuthor());

}

};

}

public class Books\_Demo {

public static void main(String[] args) {

List<Books> b = new ArrayList<>();

b.add(new Books(103, "Java Programming", "Anita"));

b.add(new Books(101, "Data Structures", "Ravi"));

b.add(new Books(105, "Java Programming", "Zoya"));

b.add(new Books(102, "Algorithms", "Meera"));

b.add(new Books(104, "Data Structures", "Amit"));

System.out.println("Original List:");

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

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Collections.sort(b);

System.out.println("Sorted by Book ID (Comparable):");

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

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

b.sort(Books.TITLE\_THEN\_AUTHOR\_COMPARATOR);

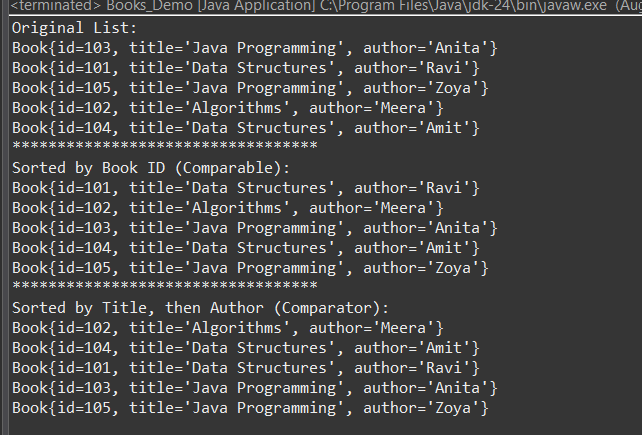
System.out.println("Sorted by Title, then Author (Comparator):");

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

}

}

**OUTPUT:**



Use Comparator.comparing() with method references to sort objects in Java 8+.

import java.util.\*;

class Product6 {

private String name;

private double price;

public Product6(String name, double price) {

this.name = name;

this.price = price;

}

public String getName() { return name; }

public double getPrice() { return price; }

*@Override*

public String toString() {

return name + " - " + price;

}

}

public class Products {

public static void main(String[] args) {

List<Product6> products = Arrays.*asList*(

new Product6("Laptop", 80000),

new Product6("Phone", 30000),

new Product6("Tablet", 20000)

);

products.sort(Comparator.*comparing*(Product6::getName));

System.***out***.println("Sorted by Name: " + products);

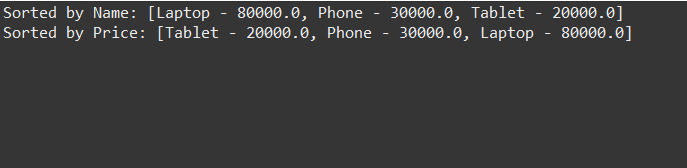
products.sort(Comparator.*comparingDouble*(Product6::getPrice));

System.***out***.println("Sorted by Price: " + products);

}

}

**OUTPUT:**



Q15. Use TreeSet with a custom comparator to sort a list of persons by age.

import java.util.\*;

class Person {

private String name;

private int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() { return name; }

public int getAge() { return age; }

*@Override*

public String toString() {

return name + " - " + age;

}

}

public class Persons3 {

public static void main(String[] args) {

TreeSet<Person> persons = new TreeSet<>(Comparator.*comparingInt*(Person::getAge));

persons.add(new Person("John", 25));

persons.add(new Person("Alice", 30));

persons.add(new Person("Bob", 22));

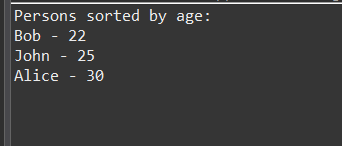
System.***out***.println("Persons sorted by age:");

persons.forEach(System.***out***::println);

}

}

**OUTPUT:**



**Create and Write to a File**

Write a Java program to create a file named student.txt and write 5 lines of student names using FileWriter.

import java.io.FileWriter;

import java.io.IOException;

public class CreateFile {

public static void main(String[] args) {

try (FileWriter fw = new FileWriter("student.txt")) {

fw.write("Ravi");

fw.write("Anush");

fw.write("Kiran");

fw.write("Priya");

fw.write("Meena");

System.out.println("File createdata written successful");

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Read from a File**

Write a program to read the contents of student.txt and display them line by line using BufferedReader.

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class ReadFile {

public static void main(String[] args) {

try (BufferedReader br = new BufferedReader(new FileReader("student.txt"))) {

String line;

while ((line = br.readLine()) != null) {

System.out.println(line);

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Append Data to a File**

Write a Java program to append a new student name to the existing student.txt file without overwriting existing data.

import java.io.FileWriter;

import java.io.IOException;

public class AppendFile {

public static void main(String[] args) {

try (FileWriter fw = new FileWriter("student.txt", true)) {

fw.write("\nSuresh");

System.***out***.println("Data appende");

} catch (IOException e) {

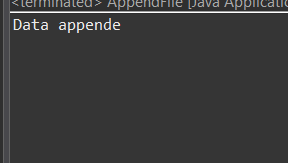
e.printStackTrace();

}

}

}

**OUTPUT:**



**Count Words and Lines**

Write a program to count the number of words and lines in a given text file notes.txt.

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class Counting {

public static void main(String[] args) {

int lines = 0, words = 0;

try (BufferedReader br = new BufferedReader(new FileReader("notes.txt "))) {

String line;

while ((line = br.readLine()) != null) {

lines++;

words += line.split("\\s+").length;

}

System.***out***.println("Lines: " + lines);

System.***out***.println("Words: " + words);

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Copy Contents from One File to Another**

Write a program to read from source.txt and write the same content into destination.txt.

import java.io.\*;

public class Copy {

public static void main(String[] args) {

try (BufferedReader br = new BufferedReader(new FileReader("source.txt"));

BufferedWriter bw = new BufferedWriter(new FileWriter("destination.txt"))) {

String line;

while ((line = br.readLine()) != null) {

bw.write(line);

bw.newLine();

}

System.***out***.println("File copied successful");

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Check if a File Exists and Display Properties**

Create a program to check if report.txt exists. If it does, display its:

* Absolute path
* File name
* Writable (true/false)
* Readable (true/false)
* File size in bytes

import java.io.File;

public class FileProps {

public static void main(String[] args) {

File file = new File("report.txt");

if (file.exists()) {

System.***out***.println("Absolute Path: " + file.getAbsolutePath());

System.***out***.println("File Name: " + file.getName());

System.***out***.println("Writable: " + file.canWrite());

System.***out***.println("Readable: " + file.canRead());

System.***out***.println("Size: " + file.length());

} else {

System.***out***.println("File does not exist");

}

}

}