1. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

// Book class implementing Comparable

class Book implements Comparable<Book> {

int bookId;

String title;

String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

// Compare books by bookId in descending order

@Override

public int compareTo(Book other) {

// Descending order: higher bookId comes first

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

}

@Override

public String toString() {

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

}

}

public class BookSortDemo {

public static void main(String[] args) {

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

// Add books

books.add(new Book(101, "Java Basics", "Alice"));

books.add(new Book(105, "Data Structures", "Bob"));

books.add(new Book(103, "Algorithms", "Charlie"));

books.add(new Book(102, "OOP Concepts", "Diana"));

// Display before sorting

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

for (Book b : books) {

System.out.println(b);

}

// Sort books by bookId descending

Collections.sort(books);

// Display after sorting

System.out.println("\nAfter sorting by bookId (descending):");

for (Book b : books) {

System.out.println(b);

}

}

}

2. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

// City class

class City {

String name;

int population;

public City(String name, int population) {

this.name = name;

this.population = population;

}

@Override

public String toString() {

return name + " (Population: " + population + ")";

}

}

public class CitySorting {

public static void main(String[] args) {

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

cities.add(new City("New York", 8419000));

cities.add(new City("Los Angeles", 3980000));

cities.add(new City("Chicago", 2716000));

cities.add(new City("Houston", 2328000));

cities.add(new City("Phoenix", 1690000));

// Original List

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

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

// Sort by population ascending

cities.sort(Comparator.comparingInt(c -> c.population));

System.out.println("\nCities Sorted by Population (Ascending):");

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

}

// Add getter for population for method reference

class CityWithGetter extends City {

public CityWithGetter(String name, int population) {

super(name, population);

}

public int getPopulation() {

return this.population;

}

}

}

3. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

// Custom class: Student

class Student implements Comparable<Student> {

int rollNo;

String name;

int marks;

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

this.rollNo = rollNo;

this.name = name;

this.marks = marks;

}

// Sort by roll number (ascending)

@Override

public int compareTo(Student other) {

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

}

@Override

public String toString() {

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

}

}

public class CustomObjectSortDemo {

public static void main(String[] args) {

List<Student> students = new ArrayList<>();

// Add some students

students.add(new Student(105, "Alice", 85));

students.add(new Student(102, "Bob", 92));

students.add(new Student(108, "Charlie", 78));

students.add(new Student(101, "Diana", 90));

// Display before sorting

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

for (Student s : students) {

System.out.println(s);

}

// Sort using Comparable (by rollNo)

Collections.sort(students);

// Display after sorting

System.out.println("\nAfter sorting by roll number:");

for (Student s : students) {

System.out.println(s);

}

}

}

4. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

// Employee class implementing Comparable

class Employee implements Comparable<Employee> {

int id;

String name;

double salary;

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

this.id = id;

this.name = name;

this.salary = salary;

}

// Compare employees by name alphabetically

@Override

public int compareTo(Employee other) {

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

}

@Override

public String toString() {

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

}

}

public class EmployeSortDemo {

public static void main(String[] args) {

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

// Add employees

employees.add(new Employee(101, "John", 50000));

employees.add(new Employee(102, "Alice", 60000));

employees.add(new Employee(103, "Bob", 55000));

employees.add(new Employee(104, "Diana", 70000));

// Display before sorting

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

for (Employee e : employees) {

System.out.println(e);

}

// Sort employees by name

Collections.sort(employees);

// Display after sorting

System.out.println("\nAfter sorting by name:");

for (Employee e : employees) {

System.out.println(e);

}

}

}

5. package Day9;

import java.io.\*;

import java.util.\*;

import java.util.Scanner;

import java.io.Serializable;

public class FileHandlingTasks {

// Q1. Create and Write to a File

public static void createAndWriteFile() {

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

writer.write("Alice\nBob\nCharlie\nDavid\nEve\n");

System.out.println("Q1: Data written to student.txt successfully.");

} catch (IOException e) { e.printStackTrace(); }

}

// Q2. Read from a File

public static void readFile() {

System.out.println("Q2: Reading student.txt:");

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(); }

}

// Q3. Append Data to a File

public static void appendToFile() {

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

writer.write("Frank\n");

System.out.println("Q3: Data appended successfully.");

} catch (IOException e) { e.printStackTrace(); }

}

// Q4. Count Words and Lines

public static void countWordsAndLines() {

int lines = 0, words = 0;

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

String line;

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

lines++;

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

}

System.out.println("Q4: Lines: " + lines + ", Words: " + words);

} catch (IOException e) { e.printStackTrace(); }

}

// Q5. Copy contents from one file to another

public static void copyFile() {

try (FileReader fr = new FileReader("source.txt");

FileWriter fw = new FileWriter("destination.txt")) {

int ch;

while ((ch = fr.read()) != -1) fw.write(ch);

System.out.println("Q5: File copied successfully.");

} catch (IOException e) { e.printStackTrace(); }

}

// Q6. Check if a File Exists and Display Properties

public static void fileProperties() {

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

if (file.exists()) {

System.out.println("Q6: 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("File Size: " + file.length() + " bytes");

} else System.out.println("Q6: File does not exist.");

}

// Q7. Create a File and Accept User Input

public static void userInputToFile() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter text to write to file:");

String input = sc.nextLine();

try (FileWriter writer = new FileWriter("userinput.txt")) {

writer.write(input);

System.out.println("Q7: Input written to userinput.txt");

} catch (IOException e) { e.printStackTrace(); }

}

// Q8. Reverse File Content

public static void reverseFileContent() {

List<String> lines = new ArrayList<>();

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

String line;

while ((line = br.readLine()) != null) lines.add(line);

} catch (IOException e) { e.printStackTrace(); }

try (BufferedWriter bw = new BufferedWriter(new FileWriter("reversed.txt"))) {

for (int i = lines.size() - 1; i >= 0; i--) {

bw.write(lines.get(i));

bw.newLine();

}

System.out.println("Q8: Reversed content written to reversed.txt");

} catch (IOException e) { e.printStackTrace(); }

}

// Q9. Store Objects in a File using Serialization

static class Student implements Serializable {

int id; String name; int marks;

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

}

public static void serializeStudent() {

Student s = new Student(1, "Alice", 90);

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("student.ser"))) {

oos.writeObject(s);

System.out.println("Q9: Student object serialized to student.ser");

} catch (IOException e) { e.printStackTrace(); }

}

// Q10. Read Serialized Object from File

public static void deserializeStudent() {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("student.ser"))) {

Student s = (Student) ois.readObject();

System.out.println("Q10: ID: " + s.id + ", Name: " + s.name + ", Marks: " + s.marks);

} catch (IOException | ClassNotFoundException e) { e.printStackTrace(); }

}

// Q11. Print All Files in a Directory

public static void listFiles() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter folder path:");

String path = sc.nextLine();

File folder = new File(path);

if (folder.exists() && folder.isDirectory()) {

File[] files = folder.listFiles();

System.out.println("Q11: Files in directory:");

for (File f : files) if (f.isFile()) System.out.println(f.getName());

} else System.out.println("Q11: Invalid folder path.");

}

// Q12. Delete a File

public static void deleteFile() {

File file = new File("file\_to\_delete.txt");

if (file.exists()) {

if (file.delete()) System.out.println("Q12: File deleted successfully.");

else System.out.println("Q12: Failed to delete the file.");

} else System.out.println("Q12: File does not exist.");

}

// Q13. Word Search in a File

public static void wordSearch() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter word to search:");

String word = sc.nextLine();

boolean found = false;

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

String line;

while ((line = br.readLine()) != null) if (line.contains(word)) { found = true; break; }

System.out.println(found ? "Q13: Word found." : "Q13: Word not found.");

} catch (IOException e) { e.printStackTrace(); }

}

// Q14. Replace a Word in a File

public static void replaceWord() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter word to replace:");

String oldWord = sc.nextLine();

System.out.println("Enter new word:");

String newWord = sc.nextLine();

File inputFile = new File("notes.txt");

File tempFile = new File("temp.txt");

try (BufferedReader br = new BufferedReader(new FileReader(inputFile));

BufferedWriter bw = new BufferedWriter(new FileWriter(tempFile))) {

String line;

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

line = line.replaceAll(oldWord, newWord);

bw.write(line); bw.newLine();

}

} catch (IOException e) { e.printStackTrace(); }

// Replace original file

if (inputFile.delete()) tempFile.renameTo(inputFile);

System.out.println("Q14: Word replacement done.");

}

// Main method to call any task

public static void main(String[] args) {

createAndWriteFile();

readFile();

appendToFile();

countWordsAndLines();

copyFile();

fileProperties();

userInputToFile();

reverseFileContent();

serializeStudent();

deserializeStudent();

listFiles();

deleteFile();

wordSearch();

replaceWord();

}

}

6. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

// Product class implementing Comparable

class Product implements Comparable<Product> {

int id;

String name;

double price;

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

this.id = id;

this.name = name;

this.price = price;

}

// Compare products by price (ascending)

@Override

public int compareTo(Product other) {

return Double.compare(this.price, other.price);

}

@Override

public String toString() {

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

}

}

public class ProductSortDemo {

public static void main(String[] args) {

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

// Add products

products.add(new Product(101, "Laptop", 55000));

products.add(new Product(102, "Phone", 20000));

products.add(new Product(103, "Tablet", 30000));

products.add(new Product(104, "Monitor", 15000));

// Display before sorting

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

for (Product p : products) {

System.out.println(p);

}

// Sort products by price (ascending)

Collections.sort(products);

// Display after sorting

System.out.println("\nAfter sorting by price:");

for (Product p : products) {

System.out.println(p);

}

}

}

7. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

// Product class

class Product {

int id;

String name;

double price;

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

this.id = id;

this.name = name;

this.price = price;

}

@Override

public String toString() {

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

}

}

public class ProductSorting {

public static void main(String[] args) {

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

products.add(new Product(101, "Laptop", 55000));

products.add(new Product(102, "Phone", 20000));

products.add(new Product(103, "Tablet", 25000));

products.add(new Product(104, "Monitor", 15000));

// Original List

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

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

// 1. Sort by Price ascending

products.sort(Comparator.comparingDouble(p -> p.price));

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

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

// 2. Sort by Price descending

products.sort(Comparator.comparingDouble(Product::getPrice).reversed());

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

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

// 3. Sort by Name alphabetically

products.sort(Comparator.comparing(p -> p.name));

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

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

}

}

// Add getter for price for method reference

class ProductWithGetter extends Product {

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

super(id, name, price);

}

public double getPrice() {

return this.price;

}

}

8. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

// Student class

class Student {

int rollNo;

String name;

int marks;

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

this.rollNo = rollNo;

this.name = name;

this.marks = marks;

}

@Override

public String toString() {

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

}

}

public class SortByMarksDescending {

public static void main(String[] args) {

List<Student> students = new ArrayList<>();

// Add students

students.add(new Student(105, "Alice", 85));

students.add(new Student(102, "Bob", 92));

students.add(new Student(108, "Charlie", 78));

students.add(new Student(101, "Diana", 90));

// Display before sorting

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

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

// Sort using Comparator (marks descending)

students.sort((s1, s2) -> Integer.compare(s2.marks, s1.marks));

// Display after sorting

System.out.println("\nAfter sorting by marks (descending):");

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

}

}

9. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.Comparator;

public class StringLengthSort {

public static void main(String[] args) {

List<String> strings = new ArrayList<>();

strings.add("Apple");

strings.add("Banana");

strings.add("Kiwi");

strings.add("Strawberry");

strings.add("Mango");

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

for (String s : strings) {

System.out.println(s);

}

// Sort using anonymous inner class

Collections.sort(strings, new Comparator<String>() {

@Override

public int compare(String s1, String s2) {

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

}

});

System.out.println("\nSorted List by Length:");

for (String s : strings) {

System.out.println(s);

}

}

}

10. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

// Student class implementing Comparable

class Student implements Comparable<Student> {

int rollNo;

String name;

int marks;

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

this.rollNo = rollNo;

this.name = name;

this.marks = marks;

}

// Compare students by rollNo (ascending)

@Override

public int compareTo(Student other) {

return this.rollNo - other.rollNo;

}

@Override

public String toString() {

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

}

}

public class StudentSort {

public static void main(String[] args) {

List<Student> students = new ArrayList<>();

// Add students

students.add(new Student(3, "Alice", 85));

students.add(new Student(1, "Bob", 90));

students.add(new Student(2, "Charlie", 78));

// Display before sorting

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

for (Student s : students) {

System.out.println(s);

}

// Sort using Comparable (rollNo ascending)

Collections.sort(students);

// Display after sorting

System.out.println("\nAfter sorting by roll number:");

for (Student s : students) {

System.out.println(s);

}

}

}

11. package Day9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

// Student class implementing Comparable to sort by name

class Student1 implements Comparable<Student> {

String name;

int marks;

public Student1(String name, int marks) {

this.name = name;

this.marks = marks;

}

@Override

public int compareTo(Student other) {

return this.name.compareTo(other.name); // sort by name

}

@Override

public String toString() {

return name + " - " + marks;

}

}

public class StudentSortingDemo {

public static void main(String[] args) {

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

students.add(new Student1("Alice", 85));

students.add(new Student1("Bob", 92));

students.add(new Student1("Charlie", 78));

students.add(new Student1("David", 90));

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

for (Student1 s : students) {

System.out.println(s);

}

// Sorting by name using Comparable

Collections.sort(students);

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

for (Student1 s : students) {

System.out.println(s);

}

}

}