Day9\_Answers:

1. package day9\_Assignment;

import java.util.ArrayList;

import java.util.Collections;

/\* 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.\*/

public class Student implements Comparable<Student>{

int roll;

String name;

double marks;

public Student(int roll, String name, double marks) {

super();

this.roll = roll;

this.name = name;

this.marks = marks;

}

*@Override*

public int compareTo(Student o) {

return Integer.*compare*(this.roll, o.roll);

}

public String toString() {

return "roll : "+roll+" Name : "+name+" Marks: "+marks;

}

public static void main(String[] args) {

ArrayList<Student> list=new ArrayList<Student>();

Student s1=new Student(11,"C",8.5);

Student s2=new Student(1,"B",8.0);

Student s3=new Student(8,"A",5.5);

list.add(s1);

list.add(s2);

list.add(s3);

Collections.*sort*(list);//internally uses overridden compareTo method

for(Student s:list) {

System.***out***.println(s);

}

}

}

Output:

roll : 1 Name : B Marks: 8.0

roll : 8 Name : A Marks: 5.5

roll : 11 Name : C Marks: 8.5

2. package day9\_Assignment;

import java.util.ArrayList;

import java.util.Collections;

/\*Create an Employee class and sort by name using Comparable.

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

public class Employee implements Comparable<Employee>{

String name;

int id;

public Employee(String name, int id) {

this.name = name;

this.id = id;

}

*@Override*

public int compareTo(Employee o) {

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

}

public String toString() {

return "Name: "+name+" id: "+id;

}

public static void main(String[] args) {

ArrayList<Employee> list=new ArrayList<Employee>();

list.add(new Employee("c",2));

list.add(new Employee("a",3));

list.add(new Employee("b",1));

Collections.*sort*(list);

for(Employee e:list) {

System.***out***.println(e);

}

}

}

Output:

Name: a id: 3

Name: b id: 1

Name: c id: 2

3. package day9\_Assignment;

import java.util.ArrayList;

import java.util.Collections;

/\*Create a Product class and sort products by price using Comparable.

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

public class Product implements Comparable<Product>{

String p\_name;

int price;

Product(String p\_name,int price){

this.p\_name=p\_name;

this.price=price;

}

*@Override*

public int compareTo(Product o) {

// **TODO** Auto-generated method stub

return Integer.*compare*(this.price, o.price);

}

public String toString() {

return "Price : "+price+" Product : "+p\_name;

}

public static void main(String[] args) {

ArrayList<Product> list=new ArrayList<Product>();

list.add(new Product("C",25));

list.add(new Product("A",2));

list.add(new Product("B",27));

Collections.*sort*(list);

for(Product p:list) {

System.***out***.println(p);

}

}

}

Output:

Price : 2 Product : A

Price : 25 Product : C

Price : 27 Product : B

4. import java.util.\*;

public class BookSort {

public static void main(String[] args) {

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

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

books.add(new Book(203, "Algorithms"));

books.add(new Book(150, "Data Structures"));

Collections.sort(books);

for (Book b : books) {

System.out.println(b.getBookId() + ": " + b.getTitle());

}

}

}

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 String getTitle() {

return title;

}

@Override

public int compareTo(Book other) {

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

}

}

Output:

203: Algorithms

150: Data Structures

101: Java Basics

5. import java.util.\*;

public class SortDisplay {

public static void main(String[] args) {

List<Product> list = Arrays.asList(

new Product("Book", 300),

new Product("Pen", 50),

new Product("Laptop", 50000)

);

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

Collections.sort(list);

System.out.println("After sorting: " + list);

}

}

class Product implements Comparable<Product> {

private String name;

private int price;

Product(String name, int price) {

this.name = name;

this.price = price;

}

@Override

public int compareTo(Product other) {

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

}

@Override

public String toString() {

return name + "(" + price + ")";

}

}

Output:

Before sorting: [Book(300), Pen(50), Laptop(50000)]

After sorting: [Pen(50), Book(300), Laptop(50000)]

6. import java.util.\*;

public class StudentSortExample {

public static void main(String[] args) {

List<Student> students = Arrays.asList(

new Student("Alice", 85),

new Student("Bob", 72),

new Student("Charlie", 90),

new Student("David", 85)

);

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

// Using a separate Comparator class

Collections.sort(students, new MarksDescendingComparator());

System.out.println("After sorting (Comparator class): " + students);

// Shuffle back for demonstration

Collections.shuffle(students);

System.out.println("Shuffled: " + students);

// Using lambda expression

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

System.out.println("After sorting (lambda): " + students);

}

}

class Student {

private String name;

private int marks;

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

public String getName() { return name; }

public int getMarks() { return marks; }

@Override

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

}

class MarksDescendingComparator implements Comparator<Student> {

@Override

public int compare(Student s1, Student s2) {

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

}

}

Output:

Before sorting: [Alice(85), Bob(72), Charlie(90), David(85)]

After sorting (Comparator class): [Charlie(90), Alice(85), David(85), Bob(72)]

Shuffled: [David(85), Bob(72), Charlie(90), Alice(85)]

After sorting (lambda): [Charlie(90), David(85), Alice(85), Bob(72)]

7. import java.util.\*;

public class ProductSort {

public static void main(String[] args) {

List<Product> p = Arrays.asList(

new Product("Laptop", 1200),

new Product("Mouse", 25),

new Product("Keyboard", 75),

new Product("Monitor", 300)

);

System.out.println("Original: " + p);

p.sort(Comparator.comparingDouble(Product::getPrice));

System.out.println("Price ↑: " + p);

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

System.out.println("Price ↓: " + p);

p.sort(Comparator.comparing(Product::getName));

System.out.println("Name α: " + p);

}

}

class Product {

private String name;

private double price;

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

String getName() { return name; }

double getPrice() { return price; }

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

}

Output:

Original: [Laptop(1200.0), Mouse(25.0), Keyboard(75.0), Monitor(300.0)]

Price ↑: [Mouse(25.0), Keyboard(75.0), Monitor(300.0), Laptop(1200.0)]

Price ↓: [Laptop(1200.0), Monitor(300.0), Keyboard(75.0), Mouse(25.0)]

Name α: [Keyboard(75.0), Laptop(1200.0), Monitor(300.0), Mouse(25.0)]

8. import java.util.\*;

public class CitySort {

public static void main(String[] args) {

List<City> cities = Arrays.asList(

new City("Tokyo", 37435191),

new City("Delhi", 29399141),

new City("Shanghai", 26317104)

);

cities.sort(Comparator.comparingInt(City::getPopulation).reversed());

cities.forEach(city -> System.out.println(city.getName() + ": " + city.getPopulation()));

}

}

class City {

private String name;

private int population;

City(String name, int population) {

this.name = name;

this.population = population;

}

public String getName() {

return name;

}

public int getPopulation() {

return population;

}

}

Output:

Tokyo: 37435191

Delhi: 29399141

Shanghai: 26317104

9. import java.nio.file.\*;

import java.io.IOException;

import java.util.List;

public class ReplaceJavaWithPython {

public static void main(String[] args) {

Path inputPath = Paths.get("story.txt");

Path outputPath = Paths.get("updated\_story.txt");

try {

List<String> lines = Files.readAllLines(inputPath);

for (int i = 0; i < lines.size(); i++) {

lines.set(i, lines.get(i).replace("Java", "Python"));

}

Files.write(outputPath, lines);

System.out.println("Replacement complete. Updated content written to updated\_story.txt.");

} catch (IOException e) {

e.printStackTrace();

}

}

}

Output:

Replacement complete. Updated content written to updated\_story.txt.

10. import java.io.\*;

import java.util.Scanner;

public class WordSearch {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String searchWord = scanner.nextLine();

boolean found = false;

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

String line;

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

if (line.contains(searchWord)) {

found = true;

break;

}

}

} catch (IOException e) {

e.printStackTrace();

}

if (found) {

System.out.println("The word \"" + searchWord + "\" exists in the file.");

} else {

System.out.println("The word \"" + searchWord + "\" does not exist in the file.");

}

}

}

Output:

Enter a word to search: Java

The word "Java" exists in the file.