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

Program :

```
package Day9
import java.util.*;
class Student implements Comparable<Student> {
    int rollNo;
    String name;
    double marks;
    Student(int rollNo, String name, double marks) {
        this.rollNo = rollNo;
        this.name = name;
        this.marks = marks;
    }
    public int compareTo(Student other) {
        return Integer.compare(this.rollNo, other.rollNo);
    }
    public String toString() {
        return rollNo + " - " + name + " - " + marks;
    }
    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();
        students.add(new Student(3, "Ram", 85));
        students.add(new Student(1, "Sita", 92));
        students.add(new Student(2, "John", 78));
        Collections.sort(students);
        students.forEach(System.out::println);
    }
}
```

Output : 1 - Sita - 92.0

2 - John - 78.0

3 - Ram - 85.0

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

Program :

```
package Day9;
import java.util.*;
class Product implements Comparable<Product> {
   String name;
   double price;
   Product(String name, double price) {
      this.name = name;
      this.price = price;
   }
   public int compareTo(Product p) {
      return Double.compare(this.price, p.price);
   }
   public String toString() {
      return name + " - " + price;
   }
   public static void main(String[] args) {
      List<Product> products = Arrays.asList(
         new Product("Laptop", 50000),
         new Product("Mouse", 500),
         new Product("Keyboard", 1500)
      );
      Collections.sort(products);
      products.forEach(System.out::println);
   }
}
```

Output : Mouse - 500.0

Keyboard - 1500.0

Laptop - 50000.0

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

Program :

```
package Day9;
import java.util.*;
class Employee implements Comparable<Employee> {
   String name;
   double salary;
   Employee(String name, double salary) {
      this.name = name;
      this.salary = salary;
   }
   public int compareTo(Employee e) {
      return this.name.compareTo(e.name);
   }
   public String toString() {
      return name + " - " + salary;
   }
   public static void main(String[] args) {
      List<Employee> employees = Arrays.asList(
         new Employee("Sita", 30000),
         new Employee("Ram", 28000),
         new Employee("Anil", 25000)
      );
      Collections.sort(employees);
      employees.forEach(System.out::println);
   }
}
```

Output : Anil - 25000.0

Ram - 28000.0

Sita - 30000.0

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

Program :

```
package Day9;
import java.util.*;
class Book implements Comparable<Book> {
   int bookId;
   String title;
   Book(int bookId, String title) {
      this.bookId = bookId;
      this.title = title;
   }
   public int compareTo(Book b) {
      return Integer.compare(b.bookId, this.bookId);
   }
   public String toString() {
      return bookId + " - " + title;
   }
   public static void main(String[] args) {
      List<Book> books = Arrays.asList(
         new Book(103, "Java"),
         new Book(101, "C++"),
         new Book(102, "Python")
      );
      Collections.sort(books);
      books.forEach(System.out::println);
   }
}
```

Output : 103 - Java

102 - Python

101 - C++

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

Program :

```
package Day9;

import java.util.*;

class CustomObject implements Comparable<CustomObject> {
    int id;
    String data;
    CustomObject(int id, String data) {
        this.id = id;
        this.data = data;
    }
    public int compareTo(CustomObject o) {
        return Integer.compare(this.id, o.id);
    }
    public String toString() {
        return id + " - " + data;
    }
    public static void main(String[] args) {
        List<CustomObject> list = Arrays.asList(
            new CustomObject(3, "C"),
            new CustomObject(1, "A"),
            new CustomObject(2, "B")
        );
        System.out.println("Before:");
        list.forEach(System.out::println);
        Collections.sort(list);
        System.out.println("\nAfter:");
        list.forEach(System.out::println);
    }
}
```

Output : Before:

3 - C

1 - A

2 – B

After:

1 - A

2 - B

3 – C

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

Program :

```
package Day9;
import java.util.*;
class Student_Desc {
   String name;
   double marks;
   Student_Desc(String name, double marks) {
      this.name = name;
      this.marks = marks;
   }
   public String toString() {
      return name + " - " + marks;
   }
   public static void main(String[] args) {
      List<Student_Desc> list = Arrays.asList(
         new Student_Desc("Sita", 85),
         new Student_Desc("Ram", 95),
         new Student_Desc("Anil", 75)
      );
```

```
        list.sort((a, b) -> Double.compare(b.marks, a.marks));
        list.forEach(System.out::println);
    }
}
```

Output : Ram - 95.0

Sita - 85.0

Anil - 75.0

7. Create multiple sorting strategies for a Product class.

Program :

```
package Day9;
import java.util.*;
class Product1 {
    String name;
    double price;
    Product1(String name, double price) {
        this.name = name;
        this.price = price;
    }
    public String toString() {
        return name + " - " + price;
    }
    public static void main(String[] args) {
        List<Product1> list = Arrays.asList(
            new Product1("Laptop", 45000),
            new Product1("Mouse", 800),
            new Product1("Monitor", 12000)
        );
        System.out.println("Price Ascending:");
        list.sort(Comparator.comparingDouble(p -> p.price));
        list.forEach(System.out::println);
```

```
        System.out.println("\nPrice Descending:");
        list.sort((a, b) -> Double.compare(b.price, a.price));
        list.forEach(System.out::println);
        System.out.println("\nName Alphabetically:");
        list.sort(Comparator.comparing(p -> p.name));
        list.forEach(System.out::println);
    }
}
```

Output : Price Ascending:

Mouse - 800.0

Monitor - 12000.0

Laptop - 45000.0

Price Descending:

Laptop - 45000.0

Monitor - 12000.0

Mouse - 800.0

Name Alphabetically:

Laptop - 45000.0

Monitor - 12000.0

Mouse - 800.0

8. Sort Employee objects by joining date using Comparator ?

Program :

```
package Day9;
import java.time.LocalDate;
import java.util.*;
class Employee1 {
    String name;
    LocalDate joiningDate;

    Employee1(String name, LocalDate joiningDate) {
```

```
        this.name = name;
        this.joiningDate = joiningDate;
    }
    public String toString() {
        return name + " - " + joiningDate;
    }
    public static void main(String[] args) {
        List<Employee1> list = Arrays.asList(
            new Employee1("Ram", LocalDate.of(2022, 5, 10)),
            new Employee1("Sita", LocalDate.of(2021, 3, 20)),
            new Employee1("John", LocalDate.of(2023, 1, 15))
        );
        list.sort(Comparator.comparing(emp -> emp.joiningDate));
        list.forEach(System.out::println);
    }
}
```

Output : Sita - 2021-03-20

Ram - 2022-05-10

John - 2023-01-15

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

Program :

```
package Day9;
import java.util.*;
class City {
    String name;
    int population;
    City(String name, int population) {
        this.name = name;
        this.population = population;
    }
```

```
    public String toString() {
        return name + " - Population: " + population;
    }
    public static void main(String[] args) {
        List<City> cities = Arrays.asList(
            new City("Delhi", 19000000),
            new City("Mumbai", 21000000),
            new City("Bangalore", 12000000)
        );
        cities.sort(Comparator.comparingInt(city -> city.population));
        cities.forEach(System.out::println);
    }
}
```

Output : Bangalore - Population: 12000000

Delhi - Population: 19000000

Mumbai - Population: 21000000

10. Use an anonymous inner class to sort a list of strings by length ?

Program :

```
package Day9;
import java.util.*;
public class String_Sort {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("Apple", "Banana", "Kiwi", "Mango");
        Collections.sort(list, new Comparator<String>() {
            public int compare(String s1, String s2) {
                return s1.length() - s2.length();
            }
        });
        list.forEach(System.out::println);
    }
```

```
}
```

Output : Kiwi

Apple

Mango

Banana

11. Create a program where: Student implements Comparable to sort by name

Use Comparator to sort by marks ?

Program :

```
package Day9;
import java.util.*;
class Student1 implements Comparable<Student1> {
   String name;
   int marks;
   Student1(String name, int marks) {
      this.name = name;
      this.marks = marks;
   }
   public int compareTo(Student1 s) {
      return this.name.compareTo(s.name);
   }
   public String toString() {
      return name + " - " + marks;
   }
   public static void main(String[] args) {
      List<Student1> students = Arrays.asList(
         new Student1("Sita", 90),
         new Student1("Ram", 85),
         new Student1("Anil", 95)
      );
      System.out.println("Sorted by Name :");
```

```
        Collections.sort(students);
        students.forEach(System.out::println);

        System.out.println("\nSorted by Marks :");
        students.sort((a, b) -> b.marks - a.marks);
        students.forEach(System.out::println);
    }
}
```

Output : Sorted by Name :

Anil - 95

Ram - 85

Sita - 90

Sorted by Marks :

Anil - 95

Sita - 90

Ram - 85

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

Program :

```
package Day9;
import java.util.*;
class Book1 implements Comparable<Book1> {
    int id;
    String title;
    String author;
    Book1(int id, String title, String author) {
        this.id = id;
        this.title = title;
        this.author = author;
    }
```

```
    public int compareTo(Book1 b) {
        return Integer.compare(this.id, b.id);
    }

    public String toString() {
        return id + " - " + title + " - " + author;
    }

    public static void main(String[] args) {
        List<Book1> books = Arrays.asList(
            new Book1(3, "Python", "Guido"),
            new Book1(1, "Java", "James"),
            new Book1(2, "C++", "Bjarne")
        );

        System.out.println("Sorted by ID:");
        Collections.sort(books);
        books.forEach(System.out::println);

        System.out.println("\nSorted by Title then Author:");
        books.sort(Comparator.comparing((Book1 b) -> b.title).thenComparing(b -> b.author));
        books.forEach(System.out::println);
    }
}
```

Output : Sorted by ID:

1 - Java - James

2 - C++ - Bjarne

3 - Python - Guido

Sorted by Title then Author:

2 - C++ - Bjarne

1 - Java - James

3 - Python - Guido

13. Write a menu-driven program to sort Employee objects by name, salary, or department using Comparator.

Program :

```
package Day9;

import java.util.*;

class Employee2 {
   String name, department;
   double salary;
   Employee2(String name, String department, double salary) {
      this.name = name;
      this.department = department;
      this.salary = salary;
   }
   public String toString() {
      return name + " - " + department + " - " + salary;
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      List<Employee2> list = Arrays.asList(
         new Employee2("Sita", "HR", 30000),
         new Employee2("Ram", "IT", 45000),
         new Employee2("Anil", "Sales", 35000)
      );

      System.out.println("Choose sorting option:\n1. Name\n2. Salary\n3. Department");
      int choice = sc.nextInt();
```

```
        switch (choice) {
            case 1 -> list.sort(Comparator.comparing(e -> e.name));
            case 2 -> list.sort(Comparator.comparingDouble(e -> e.salary));
            case 3 -> list.sort(Comparator.comparing(e -> e.department));
            default -> System.out.println("Invalid choice");
        }
        list.forEach(System.out::println);
    }
}
```

Output : Choose sorting option:

1. Name

2. Salary

3. Department

1

Anil - Sales - 35000.0

Ram - IT - 45000.0

Sita - HR - 30000.0

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

Program :

```
package Day9;
import java.util.*;
class Product2 {
    String name;
    double price;
    Product2(String name, double price) {
        this.name = name;
        this.price = price;
    }
    public String toString() {
        return name + " - " + price;
```

```
    }
    public double getPrice() {
        return price;
    }
    public static void main(String[] args) {
        List<Product2> list = Arrays.asList(
            new Product2("Laptop", 50000),
            new Product2("Phone", 25000),
            new Product2("Watch", 15000)
        );

        list.sort(Comparator.comparing(Product2::getPrice));
        list.forEach(System.out::println);
    }
}
```

Output : Watch - 15000.0

Phone - 25000.0

Laptop - 50000.0

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

Program :

```
package Day9;
import java.util.*;
class Person {
    String name;
    int age;
    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String toString() {
        return name + " - " + age;
```

```
    }
    public static void main(String[] args) {
        Set<Person> people = new TreeSet<>(Comparator.comparingInt(p -> p.age));
        people.add(new Person("Sita", 25));
        people.add(new Person("Ram", 22));
        people.add(new Person("Anil", 30));

        people.forEach(System.out::println);
    }
}
```

Output : Ram - 22

Sita - 25

Anil - 30

File Handling :

1. Create and Write to a File ?

Program :

```
package Day9;
import java.io.*;
public class WriteToFile {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("student.txt");
            writer.write("Ram\nSita\nJohn\nAnil\nRavi\n");
            writer.close();
            System.out.println("Written to student.txt successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output : Written to student.txt successfully.

2. Read from a File ?

Program :

```
package Day9;
import java.io.*;
public class ReadFile {
   public static void main(String[] args) {
      try {
         BufferedReader reader = new BufferedReader(new FileReader("student.txt"));
         String line;
         while ((line = reader.readLine()) != null) {
            System.out.println(line);
         }
         reader.close();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

Output: Ram

Sita

John

Anil

Ravi

3. Append Data to a File ?

Program :

```
package Day9;
import java.io.*;
```

```
public class AppendToFile {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("student.txt", true); // true for append mode
            writer.write("NewStudent\n");
            writer.close();
            System.out.println("Data appended to student.txt.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output : Data appended to student.txt.

4. Count Words and Lines ?

Program :

```
package Day9;
import java.io.*;
public class CountWords {
    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;
            }
            br.close();
            System.out.println("Lines: " + lines);
            System.out.println("Words: " + words);
```

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

Output: Lines: 6

Words: 6

5. Copy Contents from One File to Another?

Program :

```
package Day9;
import java.io.*;
public class CopyFile {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new FileReader("source.txt"));
            BufferedWriter writer = new BufferedWriter(new FileWriter("destination.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
                writer.write(line);
                writer.newLine();
            }
            reader.close();
            writer.close();
            System.out.println("File copied successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output : File copied successfully.

6. Check if a File Exists and Display Properties ?

Program :

```
package Day9;
import java.io.*;
public class FileProperties {
   public static void main(String[] args) {
      File file = new File("student.txt");
      if (file.exists()) {
         System.out.println("File exists.");
         System.out.println("Absolute path: " + file.getAbsolutePath());
         System.out.println("Name: " + file.getName());
         System.out.println("Writable: " + file.canWrite());
         System.out.println("Readable: " + file.canRead());
         System.out.println("Size (bytes): " + file.length());
      } else {
         System.out.println("File does not exist.");
      }
   }
}
```

Output : File exists.

Absolute path: C:\Users\k devendra\eclipse-workspace\Assignments\student.txt

Name: student.txt

Writable: true

Readable: true

Size (bytes): 35

7. Create a File and Accept User Input?

Program :

```
package Day9;
import java.io.*;
import java.util.Scanner;
```

```
public class UserInputFile {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter text to save in file: ");
        String input = sc.nextLine();

        try {
            FileWriter writer = new FileWriter("userinput.txt");
            writer.write(input);
            writer.close();
            System.out.println("Data saved to userinput.txt");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output : Enter text to save in file: Hii

Data saved to userinput.txt

8. Reverse File Content ?

Program :

```
package Day9;
import java.io.*;
import java.util.*;
public class ReverseFile {
    public static void main(String[] args) {
        try {
            List<String> lines = new ArrayList<>();
            BufferedReader reader = new BufferedReader(new FileReader("data.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
```

```
            lines.add(line);
        }
        reader.close();
        Collections.reverse(lines);
        BufferedWriter writer = new BufferedWriter(new FileWriter("reversed.txt"));
        for (String l : lines) {
            writer.write(l);
            writer.newLine();
        }
        writer.close();
        System.out.println("Content reversed to reversed.txt");
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

Output: Content reversed to reversed.txt

9. Store Objects in a File using Serialization ?

Program :

```
package Day9;
import java.io.*;
class Student4 implements Serializable {
  int id;
  String name;
  double marks;
  Student4(int id, String name, double marks) {
    this.id = id;
    this.name = name;
    this.marks = marks;
  }
```

```
}
public class SerializeStudent {
   public static void main(String[] args) {
      Student4 student = new Student4(1, "Ram", 85.5);
      try {
         ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("student.ser"));
         out.writeObject(student);
         out.close();
         System.out.println("Student object serialized to student.ser");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

Output: Student object serialized to student.ser

10. Read Serialized Object from File ?

Program :

```
package Day9;
import java.io.*;
public class DeserializeStudent {
   public static void main(String[] args) {
      try {
         ObjectInputStream in = new ObjectInputStream(new FileInputStream("student.ser"));
         Student4 student = (Student4) in.readObject();
         in.close();
         System.out.println("Deserialized Student:");
         System.out.println("ID: " + student.id);
         System.out.println("Name: " + student.name);
         System.out.println("Marks: " + student.marks);
      } catch (IOException | ClassNotFoundException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

Output : Deserialized Student:

ID: 1

Name: Ram

Marks: 85.5

11. Print All Files in a Directory ?

Program :

```
package Day9;
import java.io.*;
import java.util.Scanner;
public class PrintFiles {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter folder path: ");
        String path = sc.nextLine();

        File dir = new File(path);
        if (dir.isDirectory()) {
            File[] files = dir.listFiles();
            System.out.println("Files:");
            for (File f : files) {
                if (f.isFile()) {
                    System.out.println(f.getName());
                }
            }
        } else {
            System.out.println("Not a directory.");
```

```
        }
    }
}
```

Output: Enter folder path: C:/Users/k devendra/eclipse-workspace/Java_Selenium/src/File_Handling

Files:

AppendData.java

CopyFile.java

CreateNew_File.java

Des_data.java

De_data.java

Employee.java

fileDetails.java

Person.java

Printwriter_use.java

readFile_demo.java

Sample.txt

Sample1.txt

Serial_data.java

Ser_data.java

Se_data.java

Student.java

Test.java

write_file.java

12. Delete a File ?

Program :

```
package Day9;
import java.io.*;
import java.util.Scanner;
public class DeleteFile {
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter filename to delete: ");
        String filename = sc.nextLine();
        File file = new File(filename);
        if (file.exists()) {
            if (file.delete()) {
                System.out.println("File deleted.");
            } else {
                System.out.println("Could not delete file.");
            }
        } else {
            System.out.println("File not found.");
        }
    }
}
```

Output : Enter filename to delete: student.txt

File deleted.

13. Word Search in a File ?

Program :

```
package Day9;
import java.io.*;
import java.util.Scanner;
public class SearchWord {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter word to search: ");
        String word = sc.nextLine();
        try {
            BufferedReader br = new BufferedReader(new FileReader("copy.txt"));
            String line;
```

```
            boolean found = false;
            while ((line = br.readLine()) != null) {
                if (line.contains(word)) {
                    found = true;
                    break;
                }
            }
            br.close();
            System.out.println(found ? "Word found." : "Word not found.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output : Enter word to search: Hello

Word found.

14. Replace a Word in a File ?

Program :

```
package Day9;
import java.io.*;
import java.nio.file.*;

public class ReplaceFile {
    public static void main(String[] args) {
        try {
            String content = Files.readString(Path.of("copy.txt"));
            content = content.replaceAll("Java", "Python");

            Files.write(Path.of("updated_story.txt"), content.getBytes());
            System.out.println("Replaced 'Java' with 'Python' in updated_story.txt");
```

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

Output : Replaced 'Java' with 'Python' in updated_story.txt