DAY9_ASSIGNMENT

COMPARABLE & COMPARATOR

1). 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.

Ans)

```
package DAY9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

class Student1 {

   int id;

   String name;

   int marks;

   Student1(int id, String name,int marks) {

      this.id = id;

      this.name = name;

      this.marks=marks;

   }

   public String toString() {

      return id + " " + name +" "+marks;

   }

}

public class Comparable_SydentDetails 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(101, "nikhitha",77));
      list.add(new Student1(102, "chinni",89));
      list.add(new Student1(103, "chintu",76));
      Collections.sort(list, new Comparable_SydentDetails());
      list.forEach(System.out::println);
   }
}
```

Output:

```
101 nikhitha 77
102 chinni 89
103 chintu 76
```

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

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

Ans)

```
package DAY9;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
```

```
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 class Product_Class_Comparable {
    public static void main(String[] args) {
        List<Product> list = new ArrayList<>();
        list.add(new Product("laptop", 38000));
        list.add(new Product("phone", 15000));
        list.add(new Product("tablet", 7600));
        Collections.sort(list);
        for (Product p : list) {
            System.out.println(p);
        }
    }
}
```

Output:

```
tablet 7600.0
phone 15000.0
laptop 38000.0
```

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

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

Ans)

```
package Day_9;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
class Employee1 implements Comparable<Employee1> {
int id;
        String name;
   Employee1(String name,int id) {
   this.name = name;
   this.id = id;
   }
   public int compareTo(Employee1 E)
        {
                return this.name.compareTo(E.name);
        }
        public String toString() {
                return name + " " + id ;
   }
        }
        public class Employee_Comparable{
        public static void main(String[] args) {
             List<Employee1> list =new ArrayList<>();
             list.add(new Employee1("nikki",101));
             list.add(new Employee1("abhi",102));
             list.add(new Employee1("manasa",103));
             Collections.sort(list);
             list.forEach(System.out::println);
    }
        }
```

Output:

```
abhi 102
manasa 103
nikki 101
```

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

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

Ans)

```
package Day_9;
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 bb) {
        return Integer.compare(bb.bookId, this.bookId);
    }
    public String toString() {
        return bookId + " = " + title;
    }
}
public class BookSortDesc {
    public static void main(String[] args) {
        List<Book> books = new ArrayList<>();
        books.add(new Book(102, "selenium"));
        books.add(new Book(105, "java"));
        books.add(new Book(101, "maven"));
        Collections.sort(books);
        for (Book bb : books) {
            System.out.println(bb);
        }
    }
}
```

Output:

```
105 = java
```

```
102 = selenium
101 = maven
```

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

Ans)

```java
package DAY9;
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 Integer.compare(this.id, s.id);
   }
   public String toString() {
      return id + " = " + name;
   }
}
public class SortStudents {
   public static void main(String[] args) {
      List<Student> students = new ArrayList<>();
```

```
        students.add(new Student(2, "abhi"));
        students.add(new Student(3, "nikki"));
        students.add(new Student(1, "nitish"));
        System.out.println("Before Sorting:");
        for (Student st : students) {
            System.out.println(st);
        }
        Collections.sort(students);
        System.out.println("\nAfter Sorting:");
        for (Student st : students) {
            System.out.println(st);
        }
    }
}
```

Output:

```
Before Sorting:
2 = abhi
3 = nikki
1 = nitish
After Sorting:
1 = nitish
2 = abhi
3 = nikki
```

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

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

Ans)

```
package DAY9;
import java.util.*;
class Student2 {
    String name;
    int marks;
    Student2(String name, int marks) {
        this.name = name;
        this.marks = marks;
    }
    public String toString() {
        return name + " - " + marks;
    }
}
public class SortByMarks {
    public static void main(String[] args) {
        List<Student2> students = new ArrayList<>();
        students.add(new Student2("manasa", 65));
        students.add(new Student2("shruthi", 72));
        students.add(new Student2("shivani", 88));
        System.out.println("Before Sorting:");
        for (Student2 s : students) {
            System.out.println(s);
        }
        students.sort((s1, s2) -> Integer.compare(s2.marks, s1.marks));
        System.out.println("\nAfter Sorting by marks (desc):");
        for (Student2 s : students) {
            System.out.println(s);
        }
    }
}
```

Output:

```
Before Sorting:
manasa - 65
shruthi - 72
shivani - 88
After Sorting by marks (desc):
shivani - 88
shruthi - 72
manasa - 65
```

7. Create multiple sorting strategies for a Product class.

Implement comparators to sort by:

Price ascending

Price descending

Name alphabetically

Ans)

```
package Day_9;
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 class ProductSortExample {
    public static void main(String[] args) {
        List<Product1> products = new ArrayList<>();
        products.add(new Product1("lipstick", 550));
        products.add(new Product1("eyeliner", 200));
        products.add(new Product1("mascara", 300));
        System.out.println("Original List:");
        p for (Product1 p : products) {
            System.out.println(p);
        }
        products.sort(Comparator.comparingDouble(p -> p.price));
        System.out.println("\nSorted by Price (Ascending):");
        for (Product1 p : products) {
            System.out.println(p);
        }
        products.sort((p1, p2) -> Double.compare(p2.price, p1.price));
        System.out.println("\nSorted by Price (Descending):");
        for (Product1 p : products) {
            System.out.println(p);
        }
        products.sort(Comparator.comparing(p -> p.name));
        System.out.println("\nSorted by Name (Alphabetically):");
        for (Product1 p : products) {
            System.out.println(p);
        }
    }
}
```

Output:

```
Original List:
lipstick - ₹550.0
eyeliner - ₹200.0
mascara - ₹300.0
Sorted by Price (Ascending):
eyeliner - ₹200.0
mascara - ₹300.0
lipstick - ₹550.0
Sorted by Price (Descending):
lipstick - ₹550.0
mascara - ₹300.0
eyeliner - ₹200.0
Sorted by Name (Alphabetically):
```

```
eyeliner - ₹200.0
lipstick - ₹550.0
mascara - ₹300.0
```

8. Sort Employee objects by joining date using Comparator.

Use Comparator to sort employees based on LocalDate or Date.

Ans)

```
package DAY9;
import java.time.LocalDate;
import java.util.*;
class Employee {
    String name;
    LocalDate joiningDate;
    Employee(String name, LocalDate joiningDate) {
        this.name = name;
        this.joiningDate = joiningDate;
    }
    public String toString() {
        return name + " = " + joiningDate;
    }
}
public class SortByJoiningDate {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("manasa", LocalDate.of(2024, 2, 10)));
        employees.add(new Employee("vaishu", LocalDate.of(2023, 3, 23)));
        employees.add(new Employee("sath", LocalDate.of(2020, 8, 15)));
        employees.sort(Comparator.comparing(emp -> emp.joiningDate));
        for (Employee e : employees) {
            System.out.println(e);
        }
    }
}
```

Output:

```
sath = 2020-08-15
vaishu = 2023-03-23
manasa = 2024-02-10
```

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

Ans)

```
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;
    }
}
public class SortCities_Population {
    public static void main(String[] args) {
        List<City> cities = new ArrayList<>();
        cities.add(new City("Hyderabad", 20800000));
        cities.add(new City("Pune", 2060000));
        cities.add(new City("Bangalore", 1230000));
        cities.sort(Comparator.comparingInt(city -> city.population));
        for (City c : cities) {
            System.out.println(c);
        }
    }
}
```

Output:

```
Bangalore - 1230000
Pune - 2060000
Hyderabad – 20800000
```

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

Ans)

```
package DAY9;
import java.util.*;
public class SortStrings_Length {
    public static void main(String[] args) {
        List<String> names = new ArrayList<>();
        names.add("Apple");
        names.add("grapes");
        names.add("banana");
        Collections.sort(names, new Comparator<String>() {
            public int compare(String s1, String s2) {
                return Integer.compare(s1.length(), s2.length());
            }
        });
        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

Output:

```
Apple
grapes
banana
```

11. 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.

Ans)

```
package DAY9;
import java.util.*;
class Studentl implements Comparable<Studentl> {
    String name;
    int marks;
    Studentl(String name, int marks) {
        this.name = name;
        this.marks = marks;
    }
    public int compareTo(Studentl other) {
        return this.name.compareTo(other.name);
    }
    public String toString() {
        return name + " - " + marks;
    }
}
public class SortStudentExample {
    public static void main(String[] args) {
        List<Studentl> students = new ArrayList<>();
        students.add(new Studentl("shruthi", 65));
        students.add(new Studentl("nikki", 92));
        students.add(new Studentl("abhi", 78));
        // Sort by name using Comparable
        Collections.sort(students);
        System.out.println("Sorted by Name:");
        for (Studentl s : students) {
            System.out.println(s);
        }
        // Sort by marks using Comparator
        Collections.sort(students, new Comparator<Studentl>() {
            public int compare(Studentl s1, Studentl s2) {
                return Integer.compare(s1.marks, s2.marks);
            }
        });
        System.out.println("\nSorted by Marks:");
        for (Studentl s : students) {
            System.out.println(s);
        }
    }
}
```

Output:

```
Sorted by Name:
abhi - 78
nikki - 92
shruthi - 65
Sorted by Marks:
shruthi - 65
abhi - 78
nikki – 92
```

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

Ans)

```
package DAY9;
import java.util.*;
```

```
class LibraryItem implements Comparable<LibraryItem> {
    int id;
    String title;
    String author;
    LibraryItem(int id, String title, String author) {
        this.id = id;
        this.title = title;
        this.author = author;
    }
    public int compareTo(LibraryItem other) {
        return Integer.compare(this.id, other.id);
    }
    public String toString() {
        return id + " - " + title + " - " + author;
    }
}
public class SortLibrary {
    public static void main(String[] args) {
        List<LibraryItem> items = new ArrayList<>();
        items.add(new LibraryItem(3, "Java", "Nikhitha"));
        items.add(new LibraryItem(1, "Selenium", "Nitish"));
        items.add(new LibraryItem(2, "Maven", "Bobby"));
        Collections.sort(items);
        System.out.println("Sorted by ID:");
        for (LibraryItem item : items) {
            System.out.println(item);
        }
        Collections.sort(items, new Comparator<LibraryItem>() {
            public int compare(LibraryItem i1, LibraryItem i2) {
                int titleCompare = i1.title.compareTo(i2.title);
                if (titleCompare != 0) {
                    return titleCompare;
                }
                return i1.author.compareTo(i2.author);
            }
        });
        System.out.println("\nSorted by Title, then Author:");
        for (LibraryItem item : items) {
            System.out.println(item);
        }
    }
}
```

Output:

```
Sorted by ID:
1 - Selenium - Nitish
2 - Maven - Bobby
3 - Java - Nikhitha
Sorted by Title, then Author:
3 - Java - Nikhitha
2 - Maven - Bobby
```

1 - Selenium – Nitish

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

Ans)

```
package DAY9;

import java.util.*;

class CompanyEmployee {

    String name;

    double salary;

    String department;

    CompanyEmployee(String name, double salary, String department) {

        this.name = name;

        this.salary = salary;

        this.department = department;

    }

    public String toString() {

        return name + " = " + salary + " = " + department;

    }

}

public class CompanyEmployeeSortMenu {

    public static void main(String[] args) {

        List<CompanyEmployee> companyEmployees = new ArrayList<>();

        companyEmployees.add(new CompanyEmployee("Nikhitha", 50000,
"HR"));

        companyEmployees.add(new CompanyEmployee("Abhi", 70000, "IT"));
        companyEmployees.add(new CompanyEmployee("Manasa", 60000,
"Testor"));
        Scanner sc = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\n--- Sort Menu ---");
            System.out.println("1. Sort by Name");
            System.out.println("2. Sort by Salary");
            System.out.println("3. Sort by Department");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");
            choice = sc.nextInt();
            switch (choice) {
                case 1:
                    companyEmployees.sort(Comparator.comparing(emp ->
emp.name));
                    System.out.println("Sorted by Name:");
                    companyEmployees.forEach(System.out::println);
                    break;
                case 2:
                    companyEmployees.sort(Comparator.comparingDouble(emp ->
emp.salary));
                    System.out.println("Sorted by Salary:");
                    companyEmployees.forEach(System.out::println);
                    break;
                case 3:
                    companyEmployees.sort(Comparator.comparing(emp ->
emp.department));
                    System.out.println("Sorted by Department:");
                    companyEmployees.forEach(System.out::println);
                    break;
                case 4:
                    System.out.println("Exiting program...");
                    break;
                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        } while (choice != 4);
        sc.close();
    }
}
```

Output:

```
--- Sort Menu ---
1. Sort by Name
2. Sort by Salary
3. Sort by Department
4. Exit
Enter choice: 1
Sorted by Name:
Abhi = 70000.0 = IT
Manasa = 60000.0 = Testor
```

```
Nikhitha = 50000.0 = HR

--- Sort Menu ---

1. Sort by Name

2. Sort by Salary

3. Sort by Department

4. Exit

Enter choice: 2

Sorted by Salary:

Nikhitha = 50000.0 = HR

Manasa = 60000.0 = Testor

Abhi = 70000.0 = IT

--- Sort Menu ---

1. Sort by Name

2. Sort by Salary

3. Sort by Department

4. Exit

Enter choice: 3

Sorted by Department:

Nikhitha = 50000.0 = HR

Abhi = 70000.0 = IT

Manasa = 60000.0 = Testor

--- Sort Menu ---

1. Sort by Name

2. Sort by Salary

3. Sort by Department
```

```
4. Exit
Enter choice: 4
Exiting program...
```

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

Ans)

```
package DAY9;
import java.util.*;
class Person {
    String name;
    int age;
    Person(String name, int age) {
        this.name = name;
        this.age = age;
   }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
    public String toString() {
        return name + " - " + age;
    }
}
public class SortWithMethodReference {
    public static void main(String[] args) {
        List<Person> people = new ArrayList<>();
        people.add(new Person("Nitish", 25));
        people.add(new Person("Nikhitha", 30));
        people.add(new Person("Bobby", 22));
        people.sort(Comparator.comparing(Person::getName));
        System.out.println("Sorted by Name:");
        people.forEach(System.out::println);
        people.sort(Comparator.comparingInt(Person::getAge));
        System.out.println("\nSorted by Age:");
        people.forEach(System.out::println);
    }
}
```

Output:

```
Sorted by Name:
Bobby - 22
Nikhitha - 30
Nitish - 25

Sorted by Age:
Bobby - 22
Nitish - 25
Nikhitha – 30
```

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

Ans)

```
package Day_9;

import java.util.*;

class Citizen {

    String name;

    int age;

    Citizen(String name, int age) {

        this.name = name;

        this.age = age;

    }

    public String toString() {

        return name + " - " + age;

    }

}

public class TreeSetSortByAge {

    public static void main(String[] args) {

        Set<Citizen> citizens = new TreeSet<>(Comparator.comparingInt(c ->
c.age));

        citizens.add(new Citizen("Nikhitha", 25));

        citizens.add(new Citizen("Bobby", 30));

        citizens.add(new Citizen("Abhi", 20));

        for (Citizen c : citizens) {

            System.out.println(c);

        }
    }
}
```

Output:

```
Abhi - 20
Nikhitha - 25
Bobby – 30
```

FILE HANDLING

1. **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.

Ans)

```
package DAY9;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintStream;
public class Create_Write_File {
        public static void main(String[] args) throws IOException {
                FileWriter fw=new FileWriter("C:\\\\File_Handling/student.txt");
                fw.write("nikki\n");
                fw.write("abhi\n");
                fw.write("chinni\n");
                fw.write("chintu\n");
                fw.write("nitish\n");
                fw.close();
```

```
		System.out.println("successfully written to file");
	}
}
```

Output:

```
successfully written to file
```

2. Read from a File

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

Ans)

```
package DAY9;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
public class Read_File {
	public static void main(String[] args) throws IOException {
		try {
		BufferedReader r=new BufferedReader(new
FileReader("student.txt"));
		String line;
		while((line=r.readLine())!=null)
		{
			System.out.println(line);
		}
		r.close();
		}catch(IOException e)
```

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

Output:

```
Hello,I love java
```

3. 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.

Ans)

```
package DAY9;
import java.io.FileWriter;
import java.io.IOException;
public class AppendToFile {
   public static void main(String[] args) {
      try {
         FileWriter writer = new FileWriter("student.txt", true); // true = append
mode
         writer.write("manasa");
         writer.close();
         System.out.println("Data appended successfully.");
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

Output:

```
Data appended successfully.
```

4. Count Words and Lines

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

Ans)

```
package DAY9;
import java.io.*;
public class CountWordsLines {
    public static void main(String[] args) {
        int lineCount = 0;
        int wordCount = 0;
        try {
            BufferedReader reader = new BufferedReader(new
FileReader("student.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
                lineCount++;
                String[] words = line.split("\\s+");
                wordCount += words.length;
            }
            reader.close();
            System.out.println("Total Lines: " + lineCount);
            System.out.println("Total Words: " + wordCount);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

5. Copy Contents from One File to Another

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

Ans)

```
package DAY9;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class FileCopy {

    public static void main(String[] args) {

        try {

            FileReader fr = new FileReader("student.txt");

            BufferedReader br = new BufferedReader(fr);

            FileWriter fw = new FileWriter("sample.txt");

            String line;

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

                fw.write(line + "\n");

            }

            br.close();

            fw.close();
            System.out.println("File copied successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
File copied successfully.
```

Q6. 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**

Ans)

```
Package DAY9;
import java.io.File;
public class FileCheck {
    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("File Name: " + file.getName());
            System.out.println("Writable: " + file.canWrite());
            System.out.println("Readable: " + file.canRead());
            System.out.println("File Size (bytes): " + file.length());
        } else {
            System.out.println("File does not exist.");
        }
    }
}
```

Output:

```
File exists.
Absolute Path: C:\Users\nitis\OneDrive\Desktop\java
24\java_practice\student.txt
File Name: student.txt
Writable: true
Readable: true
File Size (bytes): 98
```

7. Create a File and Accept User Input

Accept input from the user (using Scanner) and write the input to a file named userinput.txt.

Ans)

```
Package DAY9;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
public class WriteUserInputToFile {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter text to write to userinput.txt:");
        String userInput = scanner.nextLine();
        try (FileWriter writer = new FileWriter("userinput.txt")) {
            writer.write(userInput);
            System.out.println("Successfully written to userinput.txt");
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file.");
            e.printStackTrace();
        }
        scanner.close();
    }
}
```

Output:

```
Successfully written to userinput.
```

8)Reverse File Content

Write a program to read a file data.txt and create another file reversed.txt containing the lines in reverse order.

Ans)

```
package File_Handling;
import java.io.*;
import java.util.*;
public class ReverseFileContent {
    public static void main(String[] args) {
        List<String> lines = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new
FileReader("sample.txt"))) {
```

```
            String line;

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

                lines.add(line);

            }

        } catch (IOException e) {

            System.out.println("Error reading file: " + e.getMessage());

            return;

        }

        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("Reversed file created successfully.");

        } catch (IOException e) {

            System.out.println("Error writing file: " + e.getMessage());

        }

    }

}
```

Output:

```
Reversed file created successfully.
```

9. Store Objects in a File using Serialization

Create a Student class with id, name, and marks. Serialize one object and save it in a file named student.ser.

Ans)

```
package DAY9;
import java.io.*;
class Studentt implements Serializable {
    private static final long serialVersionUID = 1L;
    int id;
    String name;
    double marks;
    public Studentt(int id, String name, double marks) {
        this.id = id;
        this.name = name;
        this.marks = marks;
    }
    public String toString() {
        return id + " - " + name + " - " + marks;
    }
}
public class SerializeStudent {
    public static void main(String[] args) {
        Studentt student = new Studentt(101, "Nikhitha", 65.5);
        try (FileOutputStream fos = new FileOutputStream("student.ser");
             ObjectOutputStream oos = new ObjectOutputStream(fos)) {
            oos.writeObject(student);
            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

Deserialize the student.ser file and display the object's content on the console.

Ans)

```
Package DAY9;
package File_Handling;
import java.io.*;
class Studenta implements Serializable {
    private static final long serialVersionUID = 1L;
    int id;
    String name;
    double marks;
    public Studenta(int id, String name, double marks) {
        this.id = id;
        this.name = name;
        this.marks = marks;
    }
    public String toString() {
        return id + " - " + name + " - " + marks;
    }
}
public class DeserializeStudent {
    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream("student.ser");
             ObjectInputStream ois = new ObjectInputStream(fis)) {
            Studenta student = (Studenta) ois.readObject();
            System.out.println("Deserialized Student object:");
            System.out.println(student);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

11. Print All Files in a Directory

Write a program to list all files (not directories) inside a folder path given by the user.

Ans)

```
Package DAY9;
import java.io.File;
import java.util.Scanner;

public class ListFilesInDirectory {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
System.out.print("Enter folder path: ");
String folderPath = sc.nextLine();

File folder = new File(folderPath);

if (folder.exists() && folder.isDirectory()) {
    File[] files = folder.listFiles();

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

    if (files != null) {
        boolean foundFile = false;
        for (File file : files) {
            if (file.isFile()) {
                System.out.println(file.getName());
                foundFile = true;
            }
        }
        if (!foundFile) {
            System.out.println("No files found in the directory.");
        }
    } else {
        System.out.println("Could not access the directory contents.");
    }
} else {
```

```
            System.out.println("Invalid folder path or not a directory.");
        }
        sc.close();
    }
}
```

Output:

```
Enter folder path: C:\\Program Files\\Java\\jdk-24\\bin
Files in directory:
api-ms-win-core-console-l1-1-0.dll
api-ms-win-core-console-l1-2-0.dll
api-ms-win-core-datetime-l1-1-0.dll
api-ms-win-core-debug-l1-1-0.dll
api-ms-win-core-errorhandling-l1-1-0.dll
api-ms-win-core-fibers-l1-1-0.dll
api-ms-win-core-file-l1-1-0.dll
api-ms-win-core-file-l1-2-0.dll
api-ms-win-core-file-l2-1-0.dll
api-ms-win-core-handle-l1-1-0.dll
api-ms-win-core-heap-l1-1-0.dll
api-ms-win-core-interlocked-l1-1-0.dll
api-ms-win-core-libraryloader-l1-1-0.dll
api-ms-win-core-localization-l1-2-0.dll
api-ms-win-core-memory-l1-1-0.dll
```

12. Delete a File

Write a program to delete a file (given by file name) if it exists.

Ans)

```
package DAY9;

import java.io.File;

import java.util.Scanner;

public class DeleteFile {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the file name (with path if needed) to delete: ");

        String fileName = sc.nextLine();

        File file = new File(fileName);

        if (file.exists()) {

            if (file.delete()) {

                System.out.println("File deleted successfully.");

            } else {

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

            }

        } else {

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

        }

        sc.close();

    }

}
```

Output:

```
Enter the file name (with path if needed) to delete:
C:\\File_Handling/newfile.txt
```

```
File deleted successfully.
```

13. Word Search in a File Ask the user to enter a word and check whether it exists in the file notes.txt.

Ans)

```
package DAY9;
import java.io.*;
import java.util.Scanner;
public class WordSearchInFile {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the word to search: ");
    String word = sc.nextLine();
    File file = new File("student2.txt");
    if (!file.exists()) {
      System.out.println("File notes.txt does not exist.");
      sc.close();
      return;
    }
    boolean found = false;
    try (BufferedReader br = new BufferedReader(new FileReader(file))) {
      String line;
      while ((line = br.readLine()) != null) {
        if (line.contains(word)) {
          found = true;
          break;
```

```
                }
            }
        } catch (IOException e) {
            System.out.println("Error reading the file.");
            e.printStackTrace();
        }
        if (found) {
            System.out.println("The word \"" + word + "\" exists in the file.");
        } else {
            System.out.println("The word \"" + word + "\" was NOT found in the file.");
        }
        sc.close();
    }
}
```

Output:

```
Enter the word to search: batch
The word "batch" exists in the file.
```

14. Replace a Word in a File Read content from story.txt, replace all occurrences of the word "Java" with "Python", and write the updated content to updated_story.txt

Ans)

```
package DAY9;
import java.io.*;
public class ReplaceWordInFile {
    public static void main(String[] args) {
        File inputFile = new File("student2.txt");
        File outputFile = new File("student.txt");
        if (!inputFile.exists()) {
            System.out.println("File story.txt does not exist.");
            return;
        }
        StringBuilder content = new StringBuilder();
        try (BufferedReader br = new BufferedReader(new FileReader(inputFile))) {
            String line;
            while ((line = br.readLine()) != null) {
                content.append(line).append(System.lineSeparator());
            }
        } catch (IOException e) {
            System.out.println("Error reading the file.");
            e.printStackTrace();
            return;
        }
        String updatedContent = content.toString().replace("Java", "Selenium");
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(outputFile)))
{
            bw.write(updatedContent);
            System.out.println("Updated content written to student.txt");
        } catch (IOException e) {
            System.out.println("Error writing to the file.");
            e.printStackTrace();
        }
    }
}
```

Output:

```
Updated content written to student.txt
```