TASK SET: Java I/O, File Handling, and Serialization

Task 1: Create a File and Write Text

Objective: Learn how to create and write to a file using `FileWriter`.

Description:

- Create a file named `info.txt`.
- Write the content: `"Java I/O is powerful!"` to the file.
- Append `"Learn Java step-by-step"` in a new line.

Expected Outcome:
`info.txt` should contain:

```
Java I/O is powerful!
Learn Java step-by-step
```

Task 2: Read from the File Created in Task 1

Objective: Learn to read text files using `BufferedReader`.

Description:

- Read and display the contents of `info.txt` line by line on the console.

Expected Outcome:
Output:

```
Java I/O is powerful!
Learn Java step-by-step
```

Task 3: Count the Number of Words in a File

Objective: Practice parsing file content.

Description:

- Read `info.txt`.
- Count and print the total number of words in the file.

Expected Outcome: `Total words: 6`

□ Task 4: Copy Content from One File to Another

Objective: Practice file-to-file operations.

Description:

- Copy all content from `info.txt` to `copy.txt`.

Expected Outcome:

- A new file `copy.txt` should contain the same text as `info.txt`.

□ Task 5: Write and Read Student Details (Serialization)

Objective: Learn object serialization and deserialization.

Description:

- Create a `Student` class with fields: `name`, `rollNo`, `marks`.
- Create an object of `Student` and write it to a file `student.ser`.
- Deserialize the object from the file and display it.

Expected Outcome:

```
Object serialized and written to student.ser
Deserialized Student:
Name: Rahul
Roll No: 101
Marks: 85
```

□ Task 6: Take Input from Keyboard and Save to File

Objective: Practice reading user input and saving it.

Description:

- Ask the user to input their name, email, and address from the keyboard.
- Save this data in a file `userdata.txt`.

Expected Outcome:
`userdata.txt` should contain the entered details in separate lines.

□ Task 7: Create a Log File with Timestamps

Objective: Use `BufferedWriter` and `Date` to generate logs.

Description:

- Each time the program runs, log the message `"Application started"` with a timestamp.
- Append the log entry to `app.log`.

Expected Outcome:

```
[2025-08-01 10:23:45] Application started
```

□ Task 8: Create a Contact Manager Using File Storage

Objective: Mini-project using file read/write operations.

Description:

- Allow user to:
    1. Add a new contact (name, phone number)
    2. Display all contacts
    3. Search contact by name
- Store all contacts in `contacts.txt`.

Expected Outcome:

- User can manage contacts across multiple program runs.

□ Task 9: Count Lines, Words, and Characters in a File

Objective: Text analytics using file I/O.

Description:

- Read a file `sample.txt`.
- Count:
    - Total lines
    - Total words
    - Total characters (excluding spaces)

Expected Output:

```
Lines: 4
Words: 30
Characters (excluding spaces): 120
```

□ Task 10: Store and Retrieve Multiple Students using Serialization

Objective: Learn how to write/read multiple serialized objects.

Description:

- Serialize and store a list of students (`ArrayList<Student>`) into `students.ser`.
- Read back the list and display all student details.

Expected Outcome:
All student records should be serialized and retrieved correctly.

□ Task 11: File Not Found Exception Handling

Objective: Implement proper file error handling.

Description:

- Try reading a file that does not exist.
- Catch the exception and display a user-friendly message.

Expected Outcome:

```
File not found. Please check the file name.
```

□ Task 12: Keyboard Input and Reverse Text

Objective: Practice reading and manipulating input.

Description:

- Read a line of input from the user.
- Reverse the text and print it.

Example:

```
Input: Java Programming
```

```
Output: gnimmargorP avaJ
```

□ Task 13: Read and Replace Words in File

Objective: Text processing using file I/O.

Description:

- Read from `data.txt`
- Replace all occurrences of `"Java"` with `"Python"` and save to `output.txt`.

Expected Outcome:

- All instances of "Java" replaced with "Python".

□ Task 14: Merge Two Files

Objective: File merge operations.

Description:

- Read content from `file1.txt` and `file2.txt`.
- Merge both into `merged.txt`.

□ Task 15: Directory Scanner

Objective: Practice with `File` class.

Description:

- Scan a given directory.
- List all files and subdirectories with their type (File/Directory) and size.