Java I/O (Input/Output) System – Complete Guide

---

1. Introduction to I/O Classes

What is Java I/O?

Java I/O (Input and Output) is used to **read data from input sources** (like keyboard, files) and **write data to output destinations** (like console, files, network).

I/O Packages:

Java I/O is primarily handled through two packages:

- `java.io` (classic stream-based I/O)
- `java.nio` (non-blocking I/O – advanced)

For now, we'll focus on `java.io`.

---

2. Understanding Java I/O Class Hierarchy

Java I/O is broadly divided into two types:

1. Byte Streams

- For handling raw binary data
- Superclasses: `InputStream` (read), `OutputStream` (write)

2. Character Streams

- For handling character data (text)
- Superclasses: `Reader` (read), `Writer` (write)

```
            java.io
               |
     ------------------------
     |                      |
InputStream            OutputStream  → Byte Streams
     |                      |
FileInputStream       FileOutputStream
BufferedInputStream   BufferedOutputStream
ObjectInputStream     ObjectOutputStream
     ↓                      ↓
Reader                  Writer          → Character Streams
     |                      |
FileReader            FileWriter
BufferedReader        BufferedWriter
```

---

3. File Handling in Java

□ Creating a File

```
import java.io.File;
import java.io.IOException;

public class FileCreateExample {
    public static void main(String[] args) {
        try {
            File file = new File("sample.txt");
            if (file.createNewFile()) {
                System.out.println("File created: " + file.getName());
            } else {
                System.out.println("File already exists.");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

---

4. Reading from Files

Using FileReader + BufferedReader

```
import java.io.*;

public class FileReadExample {
    public static void main(String[] args) {
        try {
            BufferedReader br = new BufferedReader(new
FileReader("sample.txt"));
            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

---

5. Writing to Files

Using FileWriter + BufferedWriter

```
import java.io.*;

public class FileWriteExample {
    public static void main(String[] args) {
        try {
            BufferedWriter writer = new BufferedWriter(new
FileWriter("sample.txt", true)); // append = true
            writer.write("Hello, this is a new line!\n");
            writer.close();
            System.out.println("Data written.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

---

6. Serialization in Java

What is Serialization?

Serialization is the process of **converting an object into a byte stream**, so that it can be saved to a file or transmitted.

To serialize:

- The class must implement `Serializable` interface.

Syntax:

```
class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    String name;
    int age;
}
```

---

7. Object Serialization (Write Object to File)

Example: Serialize Object

```
import java.io.*;

class Student implements Serializable {
    String name;
    int age;

    Student(String n, int a) {
        name = n;
        age = a;
    }
}

public class SerializeExample {
    public static void main(String[] args) {
        Student s1 = new Student("John", 20);
        try {
            ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream("student.ser"));
            out.writeObject(s1);
            out.close();
            System.out.println("Object serialized.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

---

8. Deserialization (Read Object from File)

```
import java.io.*;

class Student implements Serializable {
    String name;
    int age;
}

public class DeserializeExample {
    public static void main(String[] args) {
        try {
            ObjectInputStream in = new ObjectInputStream(new
FileInputStream("student.ser"));
            Student s = (Student) in.readObject();
            in.close();
            System.out.println("Name: " + s.name + ", Age: " + s.age);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

---

9. Reading from Keyboard

Using BufferedReader and InputStreamReader

```
import java.io.*;

public class KeyboardInputExample {
    public static void main(String[] args) {
        try {
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
            System.out.print("Enter your name: ");
            String name = br.readLine();
            System.out.println("Hello, " + name);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Using Scanner (Recommended)

```
import java.util.Scanner;

public class ScannerExample {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your age: ");
        int age = sc.nextInt();
        System.out.println("You are " + age + " years old.");
        sc.close();
    }
}
```

Summary Table

Operation	Class Used	Stream Type
Read Bytes	`FileInputStream`	Byte Stream
Write Bytes	`FileOutputStream`	Byte Stream
Read Characters	`FileReader`, `BufferedReader`	Character Stream
Write Characters	`FileWriter`, `BufferedWriter`	Character Stream
Serialize Object	`ObjectOutputStream`	Object Stream
Deserialize Object	`ObjectInputStream`	Object Stream
Keyboard Input	`BufferedReader`, `Scanner`	Character Stream

Real-World Use Case Example: Save Student Info

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

class Student implements Serializable {
    String name;
    int marks;
}

public class StudentDataApp {
    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        Student s = new Student();
        System.out.print("Enter Name: ");
        s.name = sc.nextLine();
        System.out.print("Enter Marks: ");
        s.marks = sc.nextInt();

        // Serialize to file
        ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("studentdata.ser"));
        oos.writeObject(s);
        oos.close();

        System.out.println("Student saved successfully.");
    }
}
```