**Assignment: (Core Java)**

**16: Java Input/Output (I/O)**

**Que.1 Streams in Java (InputStream, OutputStream)**

**Ans.1** A stream is a sequence of data.In Java, streams are used to perform input (read) and output (write) operations.

* InputStream → for reading data
* OutputStream → for writing data

**InputStream (Byte-based Input):**

* Superclass: java.io.InputStream
* Reads byte-oriented data (binary data: images, videos, audio, raw text).

import java.io.\*;

public class InputStreamExample {

public static void main(String[] args) {

try (InputStream in = new FileInputStream("input.txt")) {

int data;

while ((data = in.read()) != -1) {

System.out.print((char) data);

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**OutputStream (Byte-based Output):**

* Superclass: java.io.OutputStream
* Writes byte-oriented data.

import java.io.\*;

public class OutputStreamExample {

public static void main(String[] args) {

try (OutputStream out = new FileOutputStream("output.txt")) {

String data = "Hello, Java Streams!";

out.write(data.getBytes()); // convert string → bytes

System.out.println("Data written to file.");

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Que.2 Reading and Writing Data Using Streams**

**Ans.2 Read from file using FileInputStream**

import java.io.\*;

public class ReadExample {

public static void main(String[] args) {

try (FileInputStream fis = new FileInputStream("input.txt")) {

int data;

while ((data = fis.read()) != -1) {

System.out.print((char) data); // convert byte → char

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Writing Data using OutputStream**

import java.io.\*;

public class WriteExample {

public static void main(String[] args) {

try (FileOutputStream fos = new FileOutputStream("output.txt")) {

String message = "Hello, Yami";

fos.write(message.getBytes()); // convert string → bytes

System.out.println("Data written successfully.");

} catch (IOException e) {

e.printStackTrace();

}

}

}

**Que.3 Handling File I/O Operations**

**Ans.3** Java provides the java.io package to handle input/output (I/O) operations — such as reading, writing, creating, and deleting files.

**Classes Used for File I/O**

1. File Class (java.io.File): Represents file or directory pathnames
2. FileInputStream / FileOutputStream : Used for binary data
3. FileReader / FileWriter : Used for text data
4. BufferedReader/BufferedWriter: Improve performance by buffering data
5. PrintWriter: Used for writing formatted text

Example:

import java.io.\*;

public class CreateFileExample {

public static void main(String[] args) {

try {

File file = new File("example.txt");

if (file.createNewFile()) {

System.out.println("File created: " + file.getName());

} else {

System.out.println("File already exists.");

}

} catch (IOException e) {

e.printStackTrace();

}

}

}