Java Files :

In Java, you can perform File I/O (Input/Output) operations using the `java.io` package. To read and write to files, you typically use classes like `File`, `FileInputStream`, `FileOutputStream`, `BufferedReader`, and `BufferedWriter`. Below, I'll provide two examples for both reading from and writing to files in Java.

Reading from a File (File Input):

Example 1: Reading a Text File Line by Line

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

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

In this example, we use a `BufferedReader` to read a text file named "sample.txt" line by line.

Example 2: Reading Binary Data from a File

```java
import java.io.FileInputStream;
import java.io.IOException;

public class FileInputStreamExample {
    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream("binarydata.dat")) {
            int byteRead;
            while ((byteRead = fis.read()) != -1) {
```

```
            System.out.print((char) byteRead); // Assuming binarydata.dat contains character data
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

In this example, we use a `FileInputStream` to read binary data from a file named "binarydata.dat."

Writing to a File (File Output):

Example 1: Writing Text to a File

```java
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class FileWriterExample {
    public static void main(String[] args) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter("output.txt"))) {
            String text = "Hello, File I/O in Java!";
            writer.write(text);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

In this example, we use a `BufferedWriter` to write a text to a file named "output.txt."

Example 2: Writing Binary Data to a File

```java
import java.io.FileOutputStream;
import java.io.IOException;

public class FileOutputStreamExample {
    public static void main(String[] args) {
```

```
        try (FileOutputStream fos = new FileOutputStream("binaryoutput.dat")) {
            byte[] data = {0x48, 0x65, 0x6C, 0x6C, 0x6F}; // "Hello" in hexadecimal
            fos.write(data);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

In this example, we use a `FileOutputStream` to write binary data to a file named "binaryoutput.dat."

These examples demonstrate basic file input and output operations in Java. Remember to handle exceptions properly and close resources using try-with-resources to ensure proper resource management.