EXPERIMENT NO. 12: FILE I/O USING BYTE AND CHARACTER STREAMS

Objective

To demonstrate reading from and writing to files using both byte streams (FileInputStream and FileOutputStream) and character streams (FileReader and FileWriter) in a Java program.

Theory

Java provides two types of file I/O streams:
• **Byte Streams** – Used for reading and writing raw binary data. These include FileInputStream and FileOutputStream.
• **Character Streams** – Used for reading and writing text data. These include FileReader and FileWriter.
Using appropriate streams helps handle both text and binary files efficiently. It is important to always close file streams to avoid memory leaks.

Reading from a File using Byte Stream

```
public static void readUsingByteStream(String filePath) {

    try (FileInputStream fis = new FileInputStream(filePath)) {
        int data;
        while ((data = fis.read()) != -1) {
            System.out.print((char) data);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Writing to a File using Byte Stream

```
public static void writeUsingByteStream(String filePath, String content) {

    try (FileOutputStream fos = new FileOutputStream(filePath)) {
        fos.write(content.getBytes());
        System.out.println("Content written to file using byte stream.");
    } catch (IOException e) {
        e.printStackTrace();
    }
```

```
}
```

Reading from a File using Character Stream

```
public static void readUsingCharacterStream(String filePath) {

    try (FileReader fr = new FileReader(filePath)) {
        int data;
        while ((data = fr.read()) != -1) {
            System.out.print((char) data);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Writing to a File using Character Stream

```
public static void writeUsingCharacterStream(String filePath, String content) {

    try (FileWriter fw = new FileWriter(filePath)) {
        fw.write(content);
        System.out.println("Content written to file using character stream.");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Conclusion

This experiment demonstrates how to handle file I/O operations using both byte and character streams in Java. Choosing the right stream type based on file content (text or binary) ensures better performance and readability. Always use try-with-resources to manage file operations cleanly and avoid resource leaks.