JAC444 - Lecture 6

Java Input / Output

Segment 2 - File I/O

Objectives

Upon completion of this lecture, you should be able to:

- Examine Reader / Writer in Java
- Contrast CharacterStream and ByteStream
- Work with Buffered Stream
- Design and Develop File I/O programs

Reader vs InputStream

- **`Reader`** and **`InputStream`** define similar methods, but for different data types.

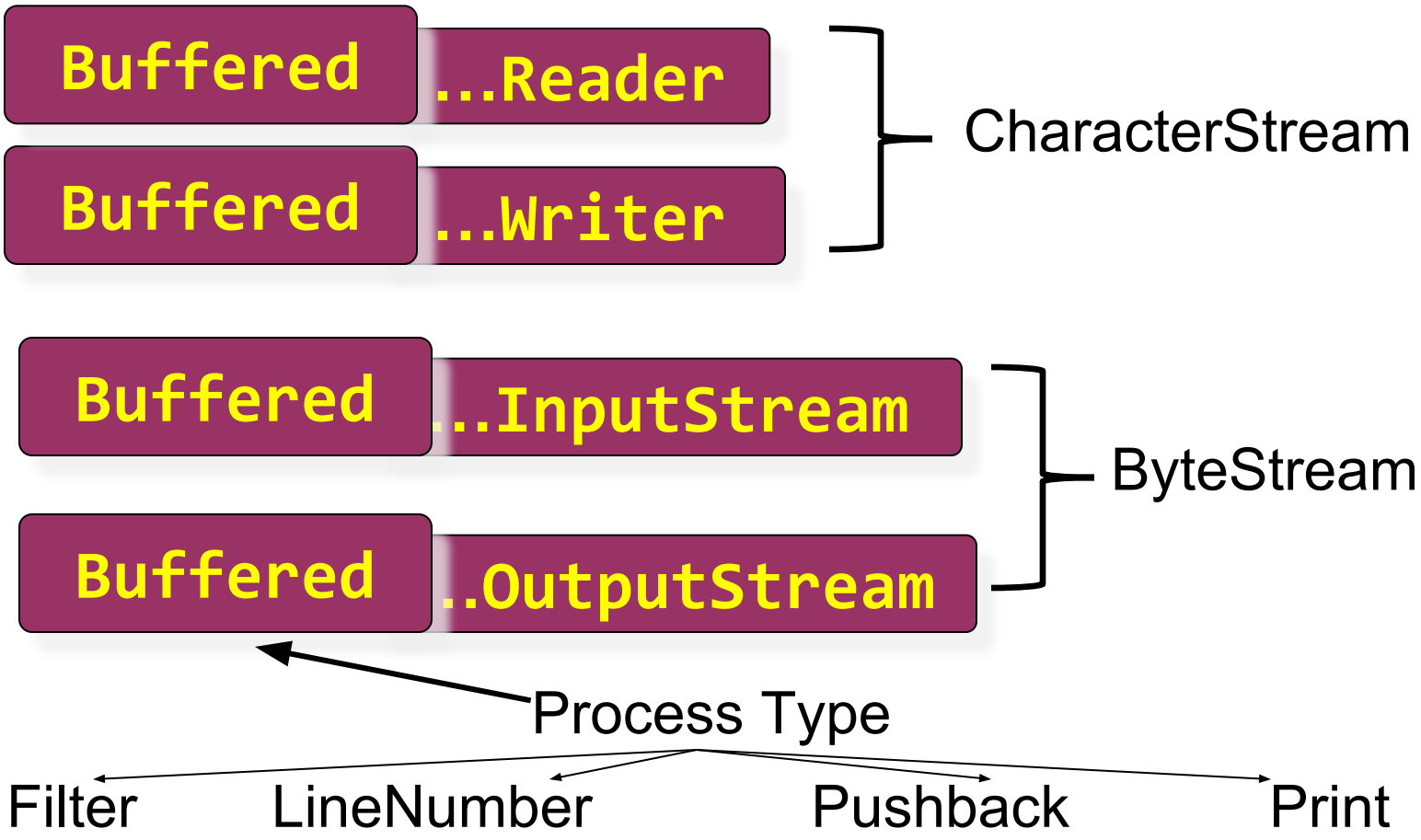
  - **`Reader`** – Reading characters and array of characters.
    - `int read()`
    - `int read(char[] cbuf)`
    - `int read(char[] cbuf, int offset, int length)`

  - **`InputStream`** – Reading bytes and array of bytes.
    - `int read()`
    - `int read(byte[] cbuf)`
    - `int read(byte[] cbuf, int offset, int length)`

File Streams Example

```
import java.io.*;

public class Copy {

   public static void main(String[] args) throws IOException {
       File inputFile = new File("args[0]");    //source
       File outputFile = new File("args[1]");   //destination

       FileReader in = new FileReader(inputFile);
       FileWriter out = new FileWriter(outputFile);
       int c;

       while ((c = in.read()) != -1)
           out.write(c);

       in.close();
       out.close();
    }
}
```

Patterns of I/O Class Names



Concatenate utility

```
import java.io.*;

public class Concatenate {

   public static void main(String[] args) throws IOException {

      ListOfFiles list = new ListOfFiles(args);

      SequenceInputStream s = new SequenceInputStream(list);
      int c;

      while ((c = s.read()) != -1)
         System.out.write(c);

      s.close();
   }
}
```

Conclusion

After completion of this segment you should know:

- How to use files in Java.
- How to read data to and write data from Java files
- Examine **`java.io`** package for IO data processing .