# CS6308
# Java Programming

V P Jayachitra

# Module V

| MODULE V I/O STREAMS | L | T | P | EL |
|---|---|---|---|---|
| | 3 | 0 | 4 | 3 |
| I/O Streams, binary I/O | | | | |
| **SUGGESTED ACTIVITIES :**<br>• Practical - binary streams, file streams<br>• EL – Lambdas and Streams | | | | |
| **SUGGESTED EVALUATION METHODS:**<br>• Assignment problems<br>• Quizzes | | | | |

# Unicode

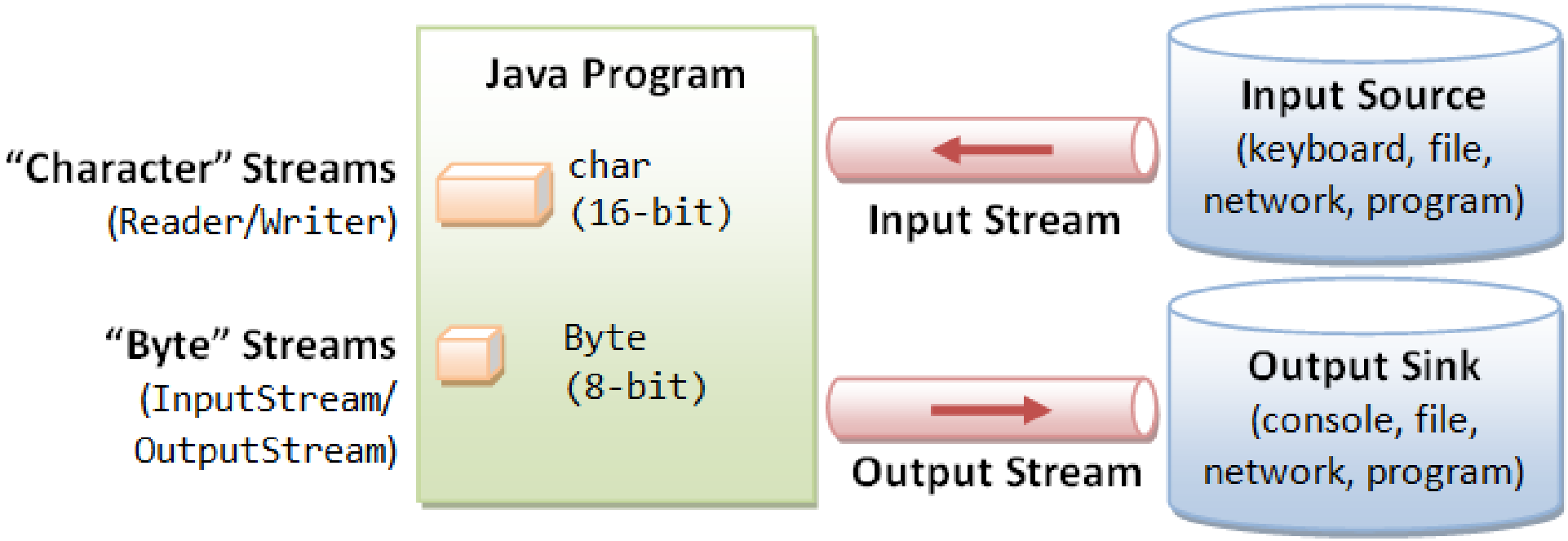
- Unicode is a character encoding
  - What : assigns a number to every character (and symbols)
  - Why : every computer prints the same character (and symbols).
- Java uses a Unicode
  - What :16 bit Unicode UTF-16
    - UTF-Unicode Transformation unit
  - Why : to unify the world language characters. i.e $2^{16}$ - 65,536 characters
  - Example : java code can contain Tamil /Chinese character(or symbol) as class name(or variable name) and string literal.

# Introduction

- **I/O**
  - Java Programs read inputs from source(eg., File) and write outputs to destination(eg. File)
  - Source or destination can also be a console, file, network, memory buffer, or another program.
- In Java standard inputs and outputs are handled by *streams*.
- A ***stream*** is a sequence of data.
  - **Input Stream :** to read data from a source, one item at a time
  - ***output stream:*** to write data to a destination, one item at time
- Stream I/O operations involve three steps:
  - *Open* an input/output stream associated with a physical device
    - (e.g., file, network, console/keyboard), by constructing an appropriate I/O stream instance.
  - *Read* from the opened input stream until "end-of-stream" encountered
  - *Write* to the opened output stream (and optionally flush the buffered output).
  - *Close* the input/output stream.

# I/O Streams

- **Byte Streams**
  - handle I/O of raw binary data.
- **Character Streams**
  - handle I/O of character data, automatic translation to and from the local character set.
- **Buffered Streams**
  - optimize input and output by reducing the number of calls to the native API.
- **Scanning and Formatting**
  - allows a program to read and write formatted text.
- **I/O from the Command Line**
  - describes the Standard Streams and the Console object.
- **Data Streams**
  - handle binary I/O of primitive data type and String values.
- **Object Streams**
  - handle binary I/O of objects.




Internal Data Formats:

- Text (char): UCS-2
- int, float, double, etc.

External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

Ref :https://www.ntu.edu.sg

# Byte streams

- Byte streams are used to read/write *raw bytes* serially from/to an external device.
- Byte streams perform input and output on 8 bits.
  - Example: the text file uses **unicode encoding** to represent character in two bytes, the byte stream will read one byte at a time.
- Byte streams should only be used for the most primitive I/O.
- read/write stream Class
  - Java.io.InputStream
  - java.io.OutputStream

# Input Stream

- Input Stream
  - **To Read from an InputStream require read() method**

```
public abstract int read() throws IOException
```

- The read() method:
  - returns the  int in the range of 0 to 255 –>byte that read
  - returns -1 -> end of stream
  - throws an IOException if it encounters an I/O error.

```
public int read(byte[] bytes, int offset, int length) throws IOException
```

```
write(bytes, 0, bytes.length)
```

```
public int read(byte[] bytes) throws IOException
```

# OutputStream

- Output Stream
  - **To write** bytes r**equire write() method**

```
public void abstract void write(int unsignedByte) throws IOException
```

  - To write a block of byte-array :

```
public void write(byte[] bytes, int offset, int length) throws IOException
```

```
example:write(bytes, 0, bytes.length)
public void write(byte[] bytes) throws IOException
```

# Character streams

- Why?
  - Easy to write programs
  - Easy to internationalize i,.e that are not dependent upon a specific character encoding.
  - Efficient than byte streams.
    - In java Char Data type is of two-byte, the only unsigned type in Java.
    - To read a single character require single read operations whereas byte stream require two read operations (char use 2 byte).
- read/write character stream class
  - java.io.Reader
  - java.io.Writer

# Reader

- superclass `Reader` operates on `char`.
- `read()` method
  - `abstract` method `read()` to read one character from the input source.
  - `read()` returns the character as an `int` between 0 to 65535
    - (a `char` in Java can be treated as an unsigned 16-bit integer);
  - or **-1** if end-of-stream is detected;
  - or throws an `IOException` if I/O error occurs.

```
public abstract int read() throws IOException
public int read(char[] chars, int offset, int length) throws IOException
public int read(char[] chars) throws IOException
```

# Writer

- superclass `Writer` operates on `char`.
- `write()` method
  - It declares an `abstract` method `write()` to `write` one character into destination.
  - or throws an `IOException` if I/O error occurs.

```
public void abstract void write(int aChar) throws IOException

public void write(char[] chars, int offset, int length) throws IOException


public void write(char[] chars) throws IOException
```

| Character-stream class | Description | Byte-stream class |
|---|---|---|
| **Reader** | input streams | **InputStream** |
| **BufferedReader** | Buffers input, parses lines | **BufferedInputStream** |
| **CharArrayReader** | Reads from a character array | **ByteArrayInputStream** |
| **InputStreamReader** | Translates byte stream into character stream (UTF-8/UTF-16 …) | **-** |
| **FileReader** | Translates bytes from a File into character stream. | **FileInputStream** |
| **Writer** | output streams | **OutputStream** |
| **BufferedWriter** | Buffers Output, uses platform's line separator | **BufferedOutputStream** |
| **CharArrayWriter** | Writes to a character array | **ByteArrayOutputStream** |
| **OutputStreamWriter** | Translates a character stream(UTF-8/UTF-16 …) into a byte stream | **-** |
| **FileWriter** | Translates character stream into a byte File | **FileOutputStream** |
| **PrintWriter** | Print in character of default Unicode. Print in bytes of default Unicode. | **PrintStream** |

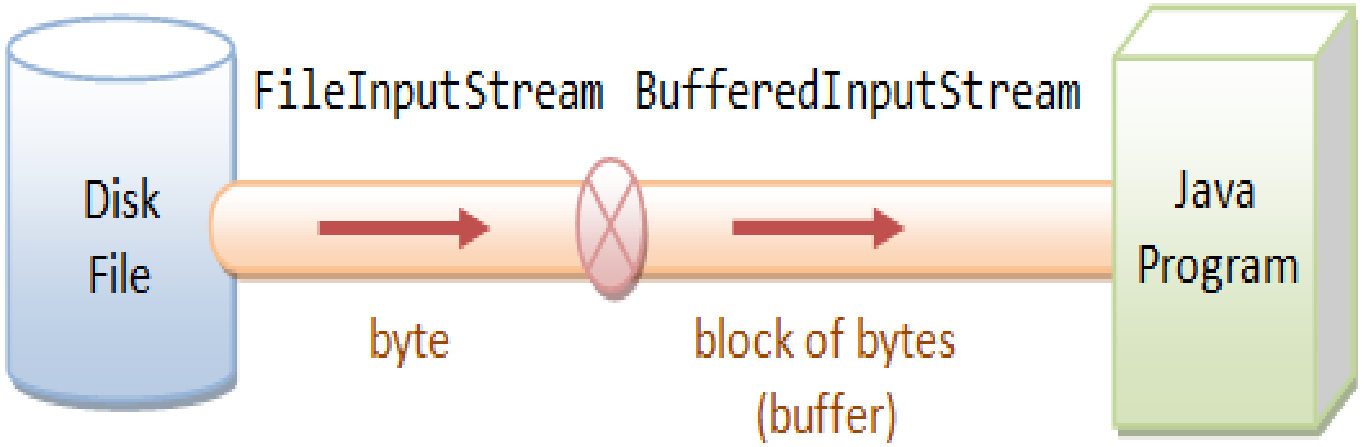
```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class CopyBytes {
        public static void main(String[] args) throws IOException {
        FileInputStream in = null;
        FileOutputStream out = null;
```

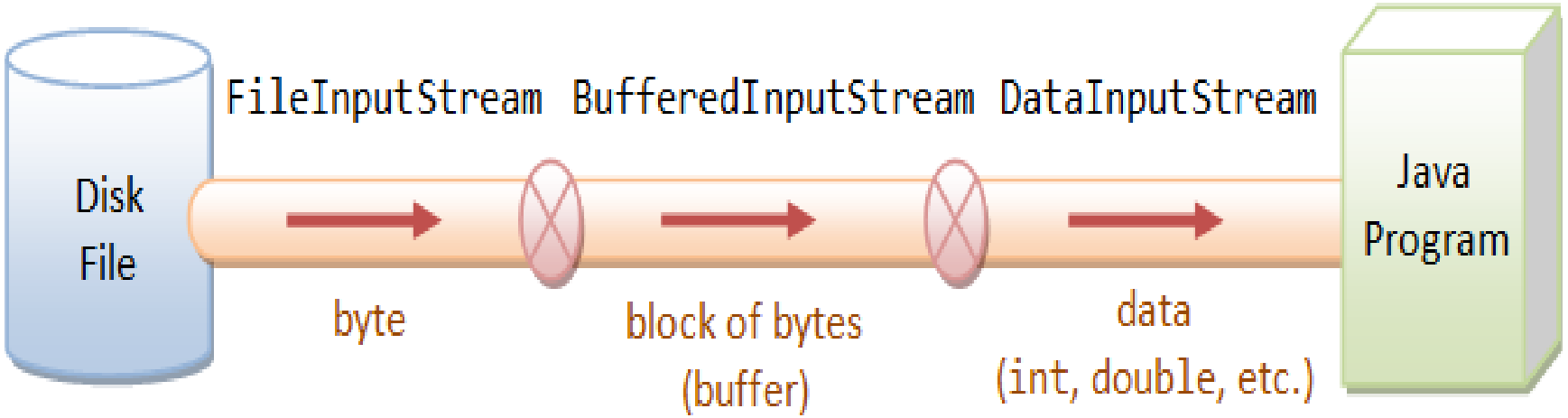
```
try {
   in = new FileInputStream(“SOURCE.txt");
  out =new FileOutputStream(“DESTINATION.txt");
  int c;
  while ((c = in.read()) != -1)
  {  out.write(c); }
  }
finally {
  if (in != null) {
      in.close(); }
  if (out != null)
   out.close(); }
}

}
}
```

OR

```
try (FileInputStream in = new
FileInputStream(“SOURCE.TXT”);
FileOutputStream out = new
FileOutputStream(“DESTINATION.TXT”))

{
int c;
while ((c = in.read()) != -1)
{ out.write(c); }
    }
catch(IOException e){ }
}
}
```

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class CopyCharacters {
        public static void main(String[] args) throws IOException {
                FileReader inputStream = null;
                FileWriter outputStream = null;
                try {
                        inputStream = new FileReader(“SOURCE.txt");
                        outputStream = new FileWriter(“DESTINATION.txt”);
                        int c;
                        while ((c = inputStream.read()) != -1) {
                                outputStream.write(c); }
                    }
                finally {
                        if (inputStream != null) {
                                inputStream.close(); }
                        if (outputStream != null) {
                                outputStream.close(); }
                         }
 } }
```




Ref :https://www.ntu.edu.sg

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class CopyBuffer{
    public static void main(String[] args) throws IOException {
try (BufferedInputStream in = new BufferedInputStream(new FileInputStream("Source.txt"));
 BufferedOutputStream out = new BufferedOutputStream(new FileOutputStream("Dest.txt")))
  {
   int c;
  while ((c = in.read()) != -1)
   { out.write(c); }
     }
 catch(IOException e) { }
}
}
```

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class CopyBuffer{
    public static void main(String[] args) throws IOException {
try (BufferedReader in = new BufferedReader(new FileReader("Source.txt"));
 BufferedWriter out = new BufferedWriter(new FileWriter("Dest.txt")))
  {
   String s;
   while ((s = in.readLine()) != null) {
   out.write(l);
     }
 catch(IOException e) { }
}
}
```

`BufferedReader` and `BufferedWriter` perform buffered I/O, instead of character-by-character.
`BufferedReader` provides a new method `readLine()`, which reads a line and returns a `String` (without the line delimiter).

Lines could be delimited by "\n"

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class CopyBuffer{
    public static void main(String[] args) throws IOException {
try (BufferedReader in = new BufferedReader(new FileReader("Source.txt"));
 PrintWriter out = new PrintWriter(new FileWriter("Dest.txt")))
 {
  String s;
  while ((s = in.readLine()) != null) {
  out.println(l);
    }
  catch(IOException e) { }
}
}
```

autoflush : PrintWriter object flushes the buffer on every invocation of println or format.
To flush a stream manually, invoke its flush method. The flush method is valid on any output stream.

```
import java.io.*;

class read{
 public static void main(String[] args) throws Exception{
// Read text file with specified unicode.
  FileInputStream fr=new FileInputStream("F:/java/eee.txt");
  InputStreamReader isr=new InputStreamReader(fr,"UTF-8");
  BufferedReader br=new BufferedReader(isr);
   String s;
  while ((s = in.readLine()) != null) {
        System.out.println(s);
}
}
}
```

```
public InputStreamReader(InputStream in) // Use default Unicode/charset
public InputStreamReader(InputStream in, String charsetName)throws UnsupportedEncodingException

public InputStreamReader(InputStream in, Charset cs)
```

InputStreamReader/OutputStreamWriter wraps in BufferedReader/BufferedWriter to read/write in multiple bytes.

# References

- https://docs.oracle.com/javase//docs/