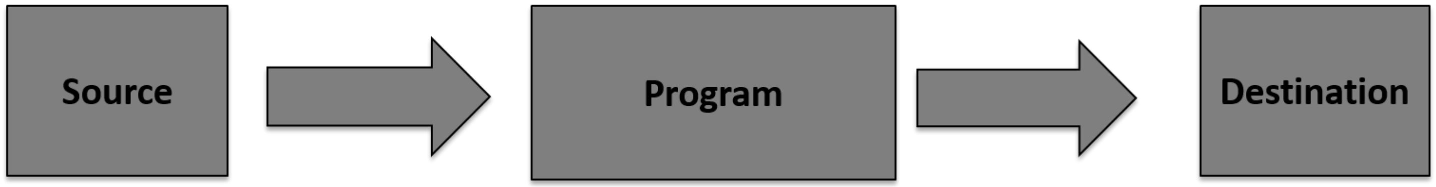
INPUT/OUTPUT

Tanjina Helaly

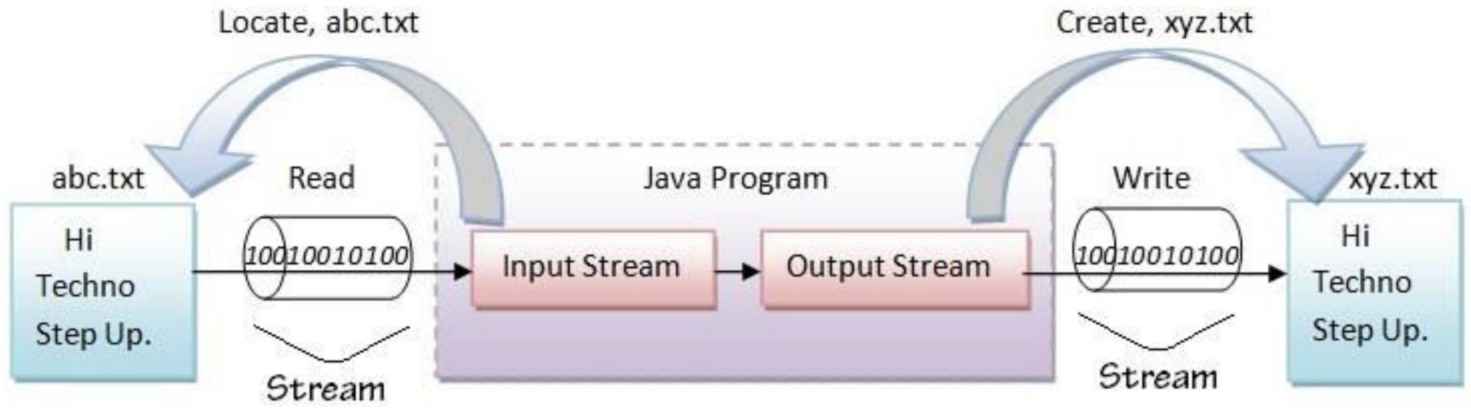
Input/Output in Java

- The java.io package - classes to perform input and output (I/O) in Java.
- All these streams represent
  - an input source and
  - an output destination.
  - supports many data such as primitives, Object, localized characters, etc.



Stream

- A stream can be defined as a sequence of data. there are two kinds of Streams
  - **InputStream:** The InputStream is used to read data from a source.
  - **OutputStream:** the OutputStream is used for writing data to a destination.



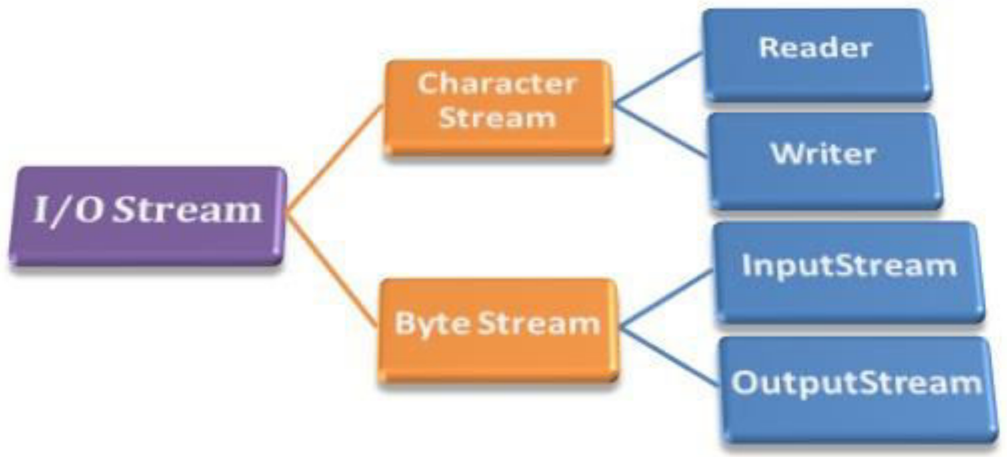
- Always close a stream when it's no longer needed
  - This practice helps avoid serious resource leaks.

I/O Stream Classifications

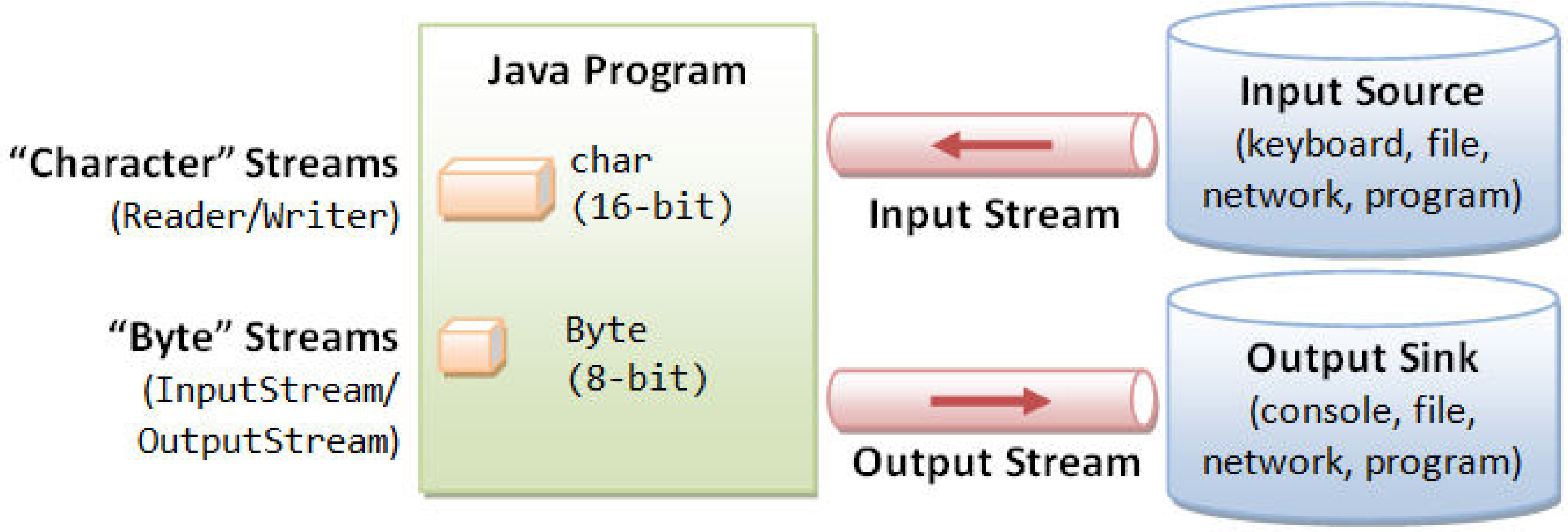
- Byte Streams handle I/O of raw binary data.
- Character Streams handle I/O of character data, automatically handling translation to and from the local character set.
- Buffered Streams optimize input and output by reducing the number of calls to the native API.
- Scanning and Formatting allows a program 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.

2 main I/O Stream

- Among the previously listed categories, two main category of IO classes are,
  - **Character- Oriented Stream:** It has two abstract classes **Reader** and **Writer**
  - **Byte- Oriented Stream:** It has two abstract classes **InputStream** and **OutputStream**



2 main I/O Stream



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)

Byte Format

- Programs use *byte streams* to perform input and output of 8-bit bytes.

Base Class	Console	File	Methods

InputStream	System.in	FileInputStream(String)
		Or
		FileInputStream(File)
			read() – return int (ASCII value of the character)

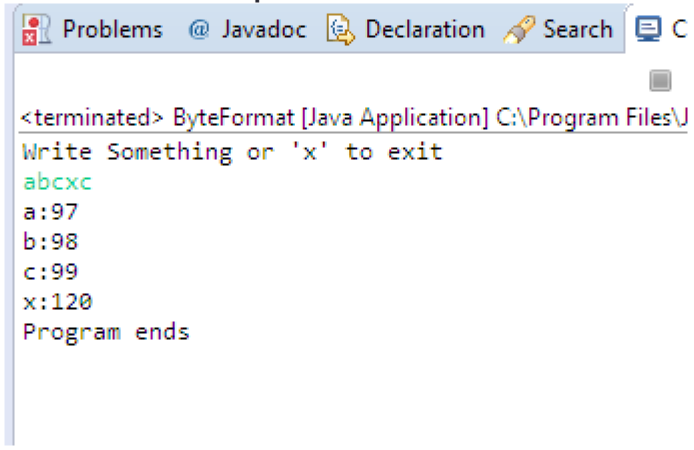
OutputStream		FileOutputStream(String
		)
		Or
		FileOutputStream(File)
			write(int)
			write(byte[])

Byte Format – From Standard Input

```
import java.io.*;

public class ByteFormat {
	public static void main(String[] args) {
		InputStream is = System.in;
		int a=0;
		System.out.println("Write Something or 'x' to
		exit");
		while (a != 'x'){
			try {
				a = is.read();
				System.out.println((char)a +":"+a);
			} catch (IOException e) {
				e.printStackTrace();
			}
		}
		try { is.close();}
		catch (IOException e) { e.printStackTrace();}
		System.out.println("Program ends");
	}
}
```

Program
Output

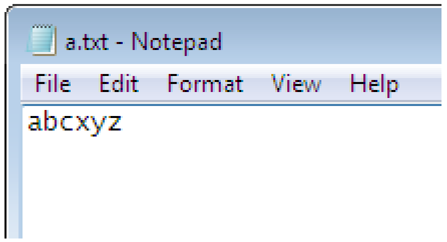


```
<terminated> ByteFormat [Java Application] C:\Program Files\J
Write Something or 'x' to exit
abcxc
a:97
b:98
c:99
x:120
Program ends
```

Byte Format – From File

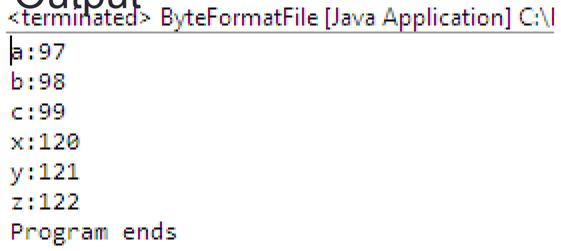
```
import java.io.*;
public class ByteFormatFile {
	public static void main(String[] args) {
		int a=0;
		FileInputStream fis;
		try {
			fis = new FileInputStream("C:\\Temp\\a.txt");
			while((a = fis.read()) != -1)
				System.out.println((char)a +":"+a);
			fis.close();
		}
		catch (FileNotFoundException e1) {
		  e1.printStackTrace();
		} catch (IOException e) {
		  e.printStackTrace();
		}
		System.out.println("Program ends");
	}
}
```

File Content



```
abcxyz
```

Program Output

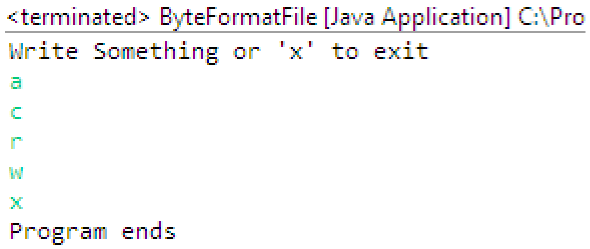


```
a:97
b:98
c:99
x:120
y:121
z:122
Program ends
```

Byte Format – Write to File

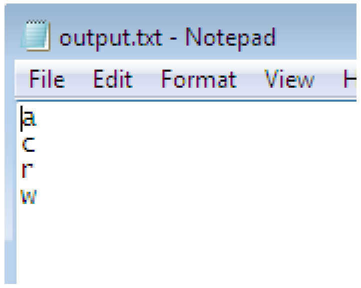
```
import java.io.*;
public class ByteFormatFile {
	public static void main(String[] args) {
		int a=0;
		FileOutputStream fos;
		InputStream is = System.in;
		try {
			fos = new FileOutputStream("C:\\Temp\\output.txt");
			System.out.println("Write Something or 'x' to exit");
			while ((a=is.read()) != 'x'){
				fos.write(a);
			}
			fos.close();
		}
		catch (FileNotFoundException e1) {
		  e1.printStackTrace();
		} catch (IOException e) {
		  e.printStackTrace();
		}
		System.out.println("Program ends");
	}
}
```

Program Output



```
<terminated> ByteFormatFile [Java Application] C:\Pro
Write Something or 'x' to exit
a
c
r
w
x
Program ends
```

File Content



```
a
c
r
w
```

Character Format

- Handle I/O of character data, automatically handling translation to and from the local character set.

Base Class	Console	File	Method
Reader	InputStreamReader(System.in)	FileReader(String) or FileReader(File)	read() – return int (ASCII character of the character)
Writer	OutputStreamWriter(System.out)	FileWriter(String) Or FileWriter(File)	write(int) write(String) append(char)

Buffered Format

- For *unbuffered* I/O.
  - each read or write request is handled directly by the underlying OS.
  - since each such request often triggers
    - disk access,
    - network activity, or
    - some other operation that is relatively expensive.
- To reduce this kind of overhead – Buffered Stream
  - Buffered **input streams read** data from a memory area known as a *buffer*;
    - the native input API is called only when the buffer is **empty**.
  - Similarly, buffered **output streams write** data to a buffer,
    - the native output API is called only when the buffer is **full**.

Buffered Format

- There are four buffered stream classes used to wrap unbuffered streams:
  - BufferedInputStream and BufferedOutputStream create buffered byte streams,
  - BufferedReader and BufferedWriter create buffered character streams.
- **Flushing Buffered Streams**
  - Sometime we need to write out a buffer at critical points, without waiting for it to fill.

Buffered Byte Format

- Handle I/O of character data, automatically handling translation to and from the local character set.

Base Class	Console	File	Method
BufferedInputStream(InputStream)	BufferedInputStream (System.in)	BufferedInputStream (FileInputStream(String or File))	read() – return int (ASCII character of the character)
BufferedOutputStream(OutputStream)	BufferedOutputStream(System.out)	BufferedOutputStream (FileOutputStream(String or File)	write(int) write(byte[])

Buffered Character Format

- Handle I/O of character data, automatically handling translation to and from the local character set.

Base Class	Console	File	Method
BufferedReader(Reader)	BufferedReader(InputStreamReader(System.in))	BufferedReader(FileReader(String or File))	read() – return int readLine()

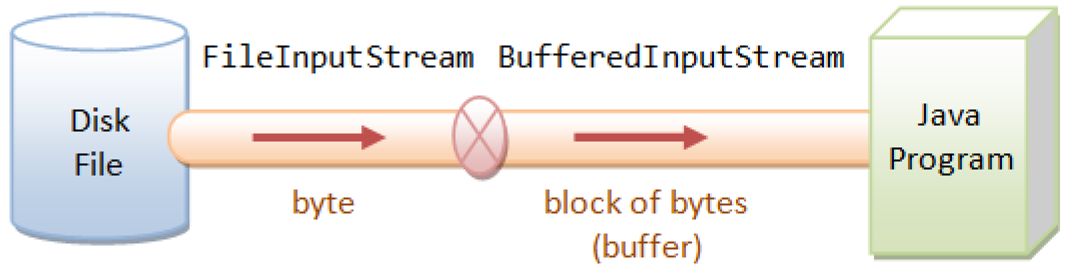
BufferedWriter(Writer)	BufferedWriter(OutputStreamWriter(System.out))	BufferedWriter(FilerWriter(String or File))	write(String)
			newLine()
			flush()

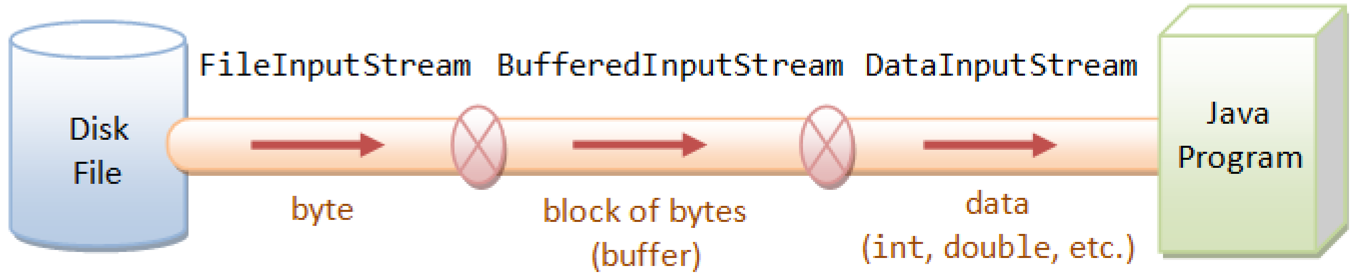
Data Stream Format

- Data streams support binary I/O of primitive data type values (boolean, char, byte, short, int, long, float, and double) as well as String values.

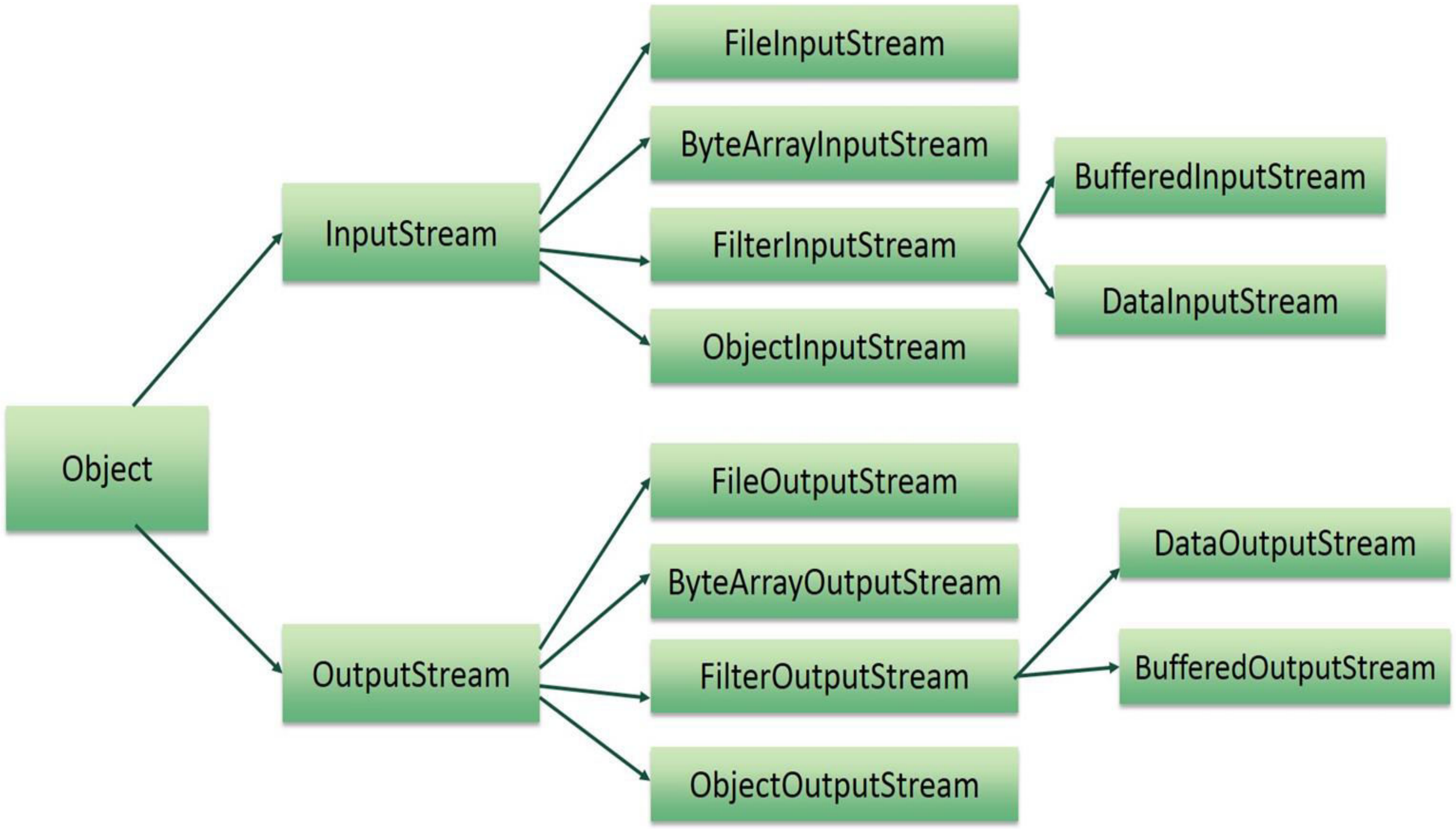
Base Class	Console	File	Method
DataInput(interface) InputStream	DataInputStream (System.in) or DataInputStream (BufferedInputStream(System.in))	DataInputStream (FileInputStream) or DataInputStream (BufferedInputStream(FileInputStream))	int i = dis.read(); boolean b = dis.readBoolean(); String s1 = dis.readLine(); String s2 = dis.readUTF();
DataOutput(Interface) OutputStream	DataOutputStream(System.out) or DataOutputStream(BufferedOutputStream(System.out))	DataOutputStream (FileOutputStream) or DataOutputStream (BufferedOutputStream(FileOutputStream))	writeInt(int); writeUTF(String); writeByte(int); writeBoolean(boolean); write(int);

Pictorial representation of different format.





Stream Hierarchy



PrintWriter

- PrintWriter contains higher level output methods to write data. Print() and println() send the toString() method to objects.
- Constructors:
  - **PrintWriter(File file)**
    - This creates a new PrintWriter, without automatic line flushing, with the specified file.
  - **PrintWriter(OutputStream out)**
    - This creates a new PrintWriter, without automatic line flushing, from an existing OutputStream.
  - **PrintWriter(OutputStream out, boolean autoFlush)**
    - This creates a new PrintWriter from an existing OutputStream
  - **PrintWriter(String fileName)**
    - This creates a new PrintWriter, without automatic line flushing, with the specified file name.
  - **PrintWriter(Writer out)**
    - This creates a new PrintWriter, without automatic line flushing.
  - **PrintWriter(Writer out, boolean autoFlush)**
    - This creates a new PrintWriter.

File I/O (Featuring NIO.2)

- The java.nio.file package
  - provide comprehensive support for file I/O and for accessing the default file system.
- Checking a File or Directory

- BufferedReader
- BufferedWriter
- PrintWriter