File Input and Output (I/O)

Similar to `fstream` in C++

For now there are **2 types of files**

- Text
  - Read using `Scanner` class
  - Written using `PrintWriter` class
- Binary
  - Read using `FileInputStream`, `DataInputStream`
  - Written using `FileOutputStream`, `DataOutputStream`

Writing text to file (PrintWriter)

Import libraries

```
import java.io.*;
```

Create an instance of `PrintWriter` class

```
// We pass the file name into the constructor
PrintWriter outFile = new PrintWriter("test.txt");
```

Write to file using `print` or `println` methods

```
outFile.println("Hello how are you?");
```

Closing the file when done

```
outFile.close();
```

Note: `PrintWriter` will throw an `IOException (FileNotFoundException)`

It is a **checked** exception, must be handled!

Prevent File Truncation

Create a `FileWriter` object

```
FileWriter fw = new FileWriter("test.txt", true);
```

Then create a `PrintWriter` object

```
PrintWriter outFile = new PrintWriter(fw);
```

PrintWriter Methods

java.io.PrintWriter	
+PrintWriter(filename: String)	Creates a PrintWriter for the specified file name.
+print(s: String): void	Writes a string.
+print(c: char): void	Writes a character.
+print(cArray: char[]): void	Writes an array of character.
+print(i: int): void	Writes an int value.
+print(l: long): void	Writes a long value.
+print(f: float): void	Writes a float value.
+print(d: double): void	Writes a double value.
+print(b: boolean): void	Writes a boolean value.
Also contains the overloaded println & printf methods.	

Reading Data using Scanner

Create `File` object

```
File myFile = new File("MyDir/MyText.txt");
```

Create `Scanner` object

```
Scanner inputFile = new Scanner(myFile);
// Note: FileNotFoundException will be thrown (checked)
```

Scanner Methods

java.util.Scanner	
+Scanner(source: File)	Creates a Scanner that produces values scanned from the specified file.
+Scanner(source: String)	Creates a Scanner that produces values scanned from the specified string.
+close()	Closes this scanner.
+hasNext(): boolean	Returns true if this scanner has another token in its input.
+next(): String	Returns next token as a string.
+nextByte(): byte	Returns next token as a byte.
+nextShort(): short	Returns next token as a short.
+nextInt(): int	Returns next token as an int.
+nextLong(): long	Returns next token as a long.
+nextFloat(): float	Returns next token as a float.
+nextDouble(): double	Returns next token as a double.
+useDelimiter(pattern: String): Scanner	Sets this scanner's delimiting pattern.

`hasNext()` Method

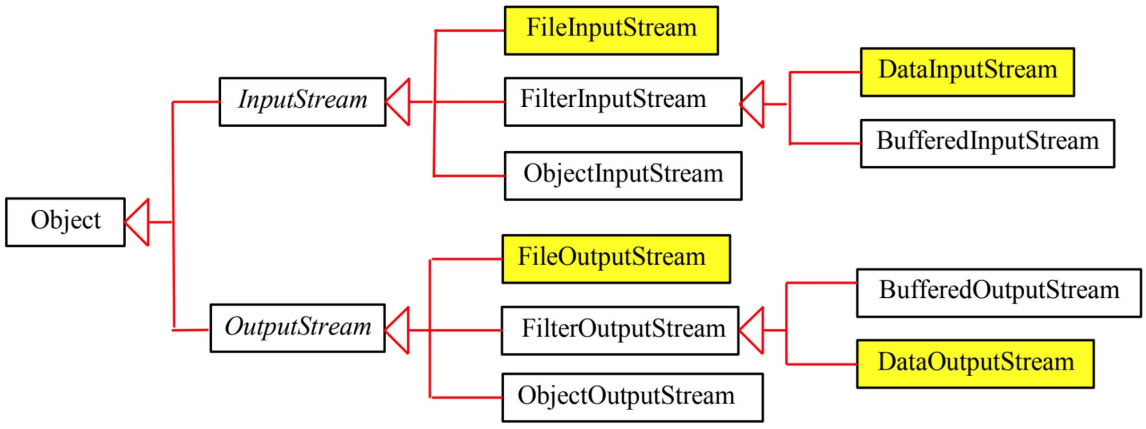
It is a boolean flag similar to `eof()` in C++

```
// File processing

while (inputFile.hasNext()) {
    String str = inputFile.nextLine();
    // ....
}

inputFile.close();
```

Binary Files

Binary I/O does not require encoding/decoding unlike Text I/O

Binary I/O Classes



Binary File Output

`FileOutputStream`

- Allows you to open a file for writing bin. data
- Establishes a connection with the file
- Provides the basic functionality of writing `bytes`

```
//Constructors
public FileOutputStream(String filename)
public FileOutputStream(File file)
public FileOutputStream(String filename, boolean append)
public FileOutputStream(File file, boolean append)

// A new will be created if it does not exist in the first place
```

`DataOutputStream`

- Allows to write any primitive type or `String` to bin. file
- Used with the above (acts as a wrapper)

```
FileOutputStream fstream = new FileOutputStream("MyInfo.dat");
DataOutputStream dstream = new DataOutputStream(fstream);

dstream.wrtieInt(7);
dstream.writeDouble(7.5);
dstream.writeChar('m');
dstream.writeChars("This is a test");
dstream.writeUTF("This is a test"); //UTF-8
dstream.writeBoolean(false);
dstream.writeByte(7);
```

Binary File Input

`FileInputStream`

```
//Constructors
public FileInputStream(String filename)
public FileInputStream(File file)
```

Note: `FileNotFoundException` might occur, must handle

`DataInputStream`

Acts as a wrapper around the above class

```
FileInputStream fstream = new FileInputStream("MyInfo.dat");
DataInputStream dstream = new DataInputStream(fstream)

// Reading primitive types
int i = dstream.readInt();
double d = dstream.readDouble();
char c = dstream.readChar();
String s = dstream.readUTF();
boolean bo = dstream.readBoolean();
byte b = dstream.readByte();
```

Reminders for Bin. IO

If you had written to file using `writeUTF()`, then you must read using `readUTF()`

Checing for EOF using the `EOFException`

```
while  (continueReading) {
    try {
        numbers = dstream.readInt();
    } catch (EOFException e) {
        continueReading = false;
    }
}
```