IO Programming

By Rahul Barve

Objectives

- Introduction to Streams.
- Understanding I/O Hierarchy.
- Stream Types.
- Understanding Serialization
- NIO Basics

Introduction to Streams

By Rahul Barve

Introduction to Streams

- Every application needs to read and write the data.
- To read the data, application requires some source and to write into, it requires some destination.

Introduction to Streams

Introduction to Streams

- An application can read the data from several sources like File, Input Device, Socket and so on.
- Similarly it can write the data into some destination like File, Output Device, Socket and so on.

Introduction to Streams

- It is possible by having a connector in between the application and the source or the destination.
- This connector is known as a stream.

Introduction to Streams

- At the base, Java library provides two types of streams: `InputStream` and `OutputStream`.
- Java provides IO library support using a package `java.io`.

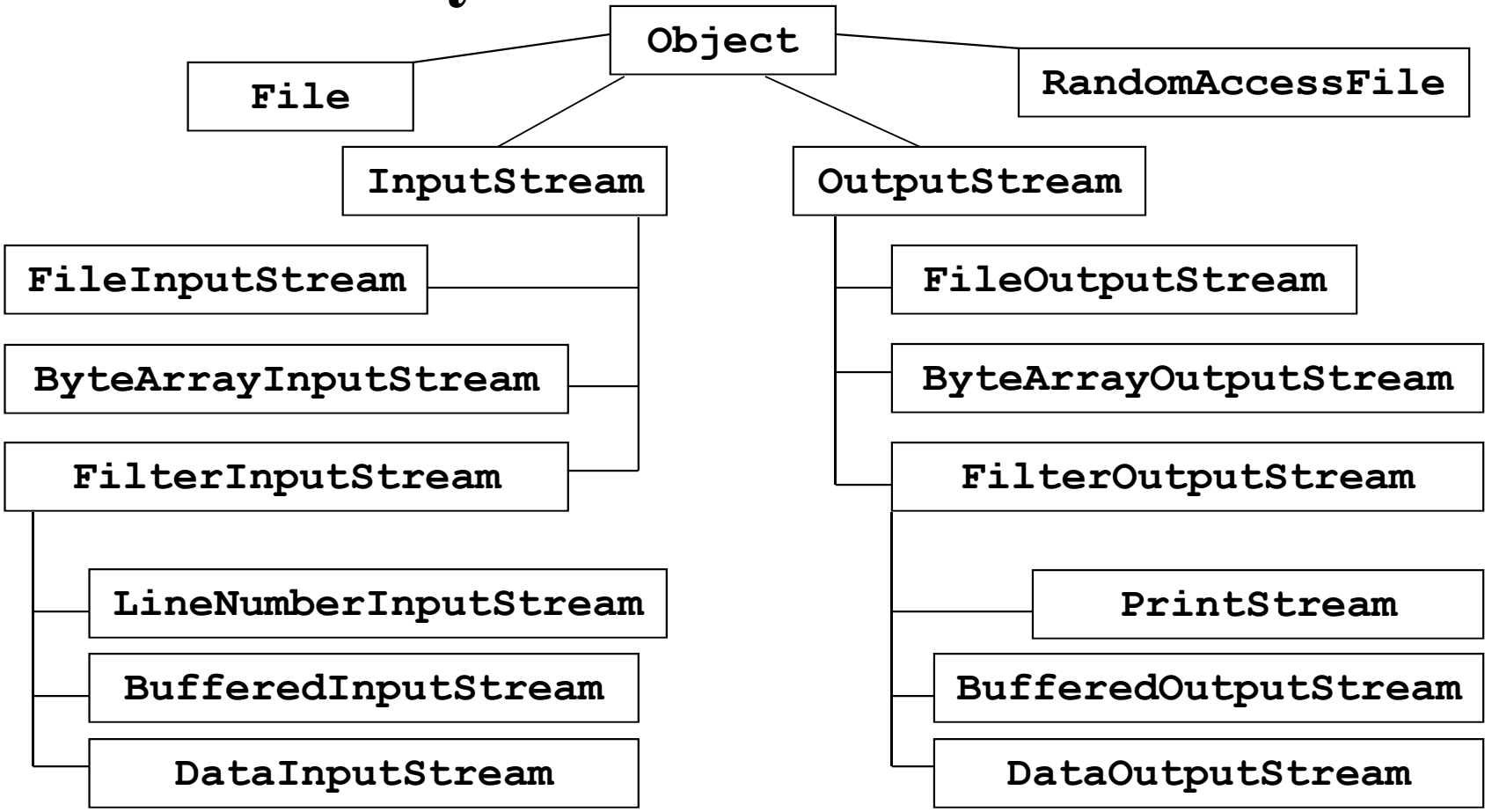
Introduction to Streams

- `InputStream` – This depicts the flow of bytes from data source to the program's memory.
- `OutputStream` – This depicts the flow of bytes from program's memory to the destination data store.

I/O Hierarchy

By Rahul Barve

I/O Hierarchy



I/O Hierarchies

By Rahul Barve

I/O Hierarchies

- There are 2 types of I/O Hierarchies available in Java
  - Byte Streams
  - Character Streams

I/O Hierarchies

- Byte streams work upon 8 bit data and fall under `InputStream` and `OutputStream` hierarchies.
- Character streams work upon 16 bit data and fall under `Reader` and `Writer` hierarchies.

InputStream

By Rahul Barve

InputStream

- An abstract base class that defines methods for performing READ operations.
- Defines a basic interface for reading streamed bytes of information.

FileInputStream

By Rahul Barve

FileInputStream

- Used to read contents of a file.
- E.g.

```
FileInputStream fin;
String file = "message.txt";
fin = new FileInputStream(file);
```

try-with-resources

By Rahul Barve

try-with-resources

- A feature introduced since Java 1.7.
- A `try` statement that declares one or more resources.

try-with-resources

- A resource is an object that must be closed after the program is finished with it.
- Any object that implements `java.lang.AutoCloseable` can be used as a resource.

try-with-resources

- E.g.

```
try (FileInputStream fin = …..){
        //Statements
}
catch(…..){
    //Statements
}
```

OutputStream

By Rahul Barve

OutputStream

- An abstract base class that defines methods for performing WRITE operations.
- Defines a basic interface for writing streamed bytes.

FileOutputStream

By Rahul Barve

FileOutputStream

- Used to write the contents to a file.
- E.g.

```
String file = "message.txt";
FileOutputStream fout;
fout = new FileOutputStream(file);
```

`FileOutputStream`

- Use 2 parameterized constructor with 2nd parameter as `boolean true` to open the file in `append` mode.

FilterInputStream

By Rahul Barve

`FilterInputStream`

- A base class for all other classes that act like a filter to transform the raw bytes to a desired form.
- Must be used through one of the sub classes.

`FilterInputStream` Subclasses

- `SequenceInputStream`
- `LineNumberInputStream`
- `BufferedInputStream`
- `DataInputStream`

`FilterInputStream` Subclasses

- `SequenceInputStream`
  - Used to represent a sequential input stream.
- `LineNumberInputStream`
  - Used to operate on the basis of line numbers.
- `BufferedInputStream`
  - Used to populate an input stream on the top of buffer.
  - Improves performance.
- `DataInputStream`
  - Used to read Java's basic primitive data types.