Core Java

IO Streams & Introducing GUI Programming

Objective

At the end of this session, you will be able to:

- Write programs using Byte Streams
- Serialize Objects
- Externalize Objects
- Write programs using Character Streams
- Introduce AWT & Swings
- Introduce Applets

Agenda

- Introduction to I/O Streams
- Types of Streams
- Byte Streams
- Object Serialization
- Object Externalization
- Character Streams
- Introduction to AWT & Swings
- Introduction to Applets

Introduction to Streams

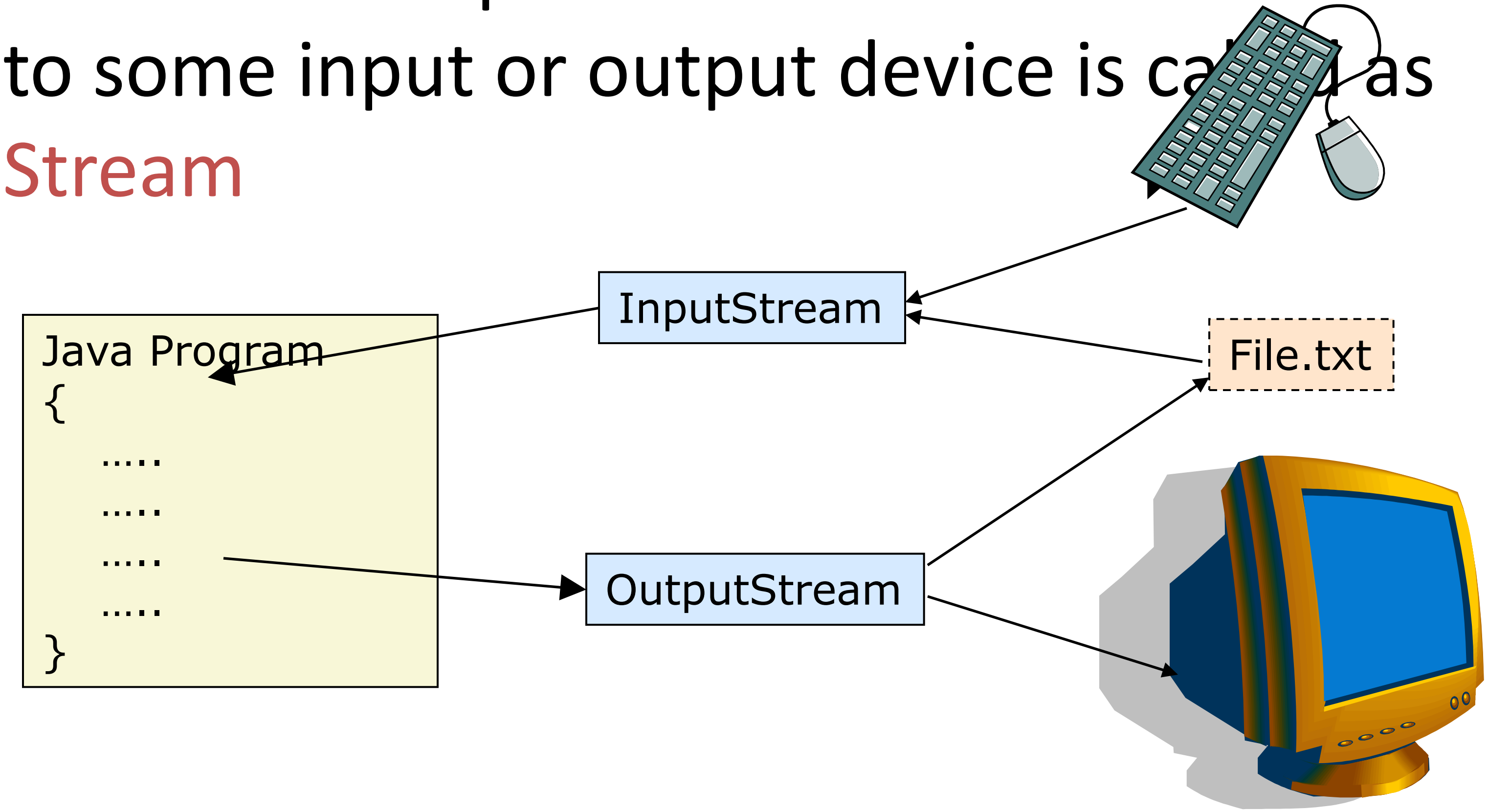
How do we take an input from keyboard?

- Input Streams are used to read data from any data source like keyboard, socket, file etc.

- Output Streams are used to write data to any data destination like console, socket, file etc.

What is I/O Stream?

- An abstract representation of data connected to some input or output device is called as Stream



Java Program
{
.....
.....
.....
.....
}

InputStream

OutputStream

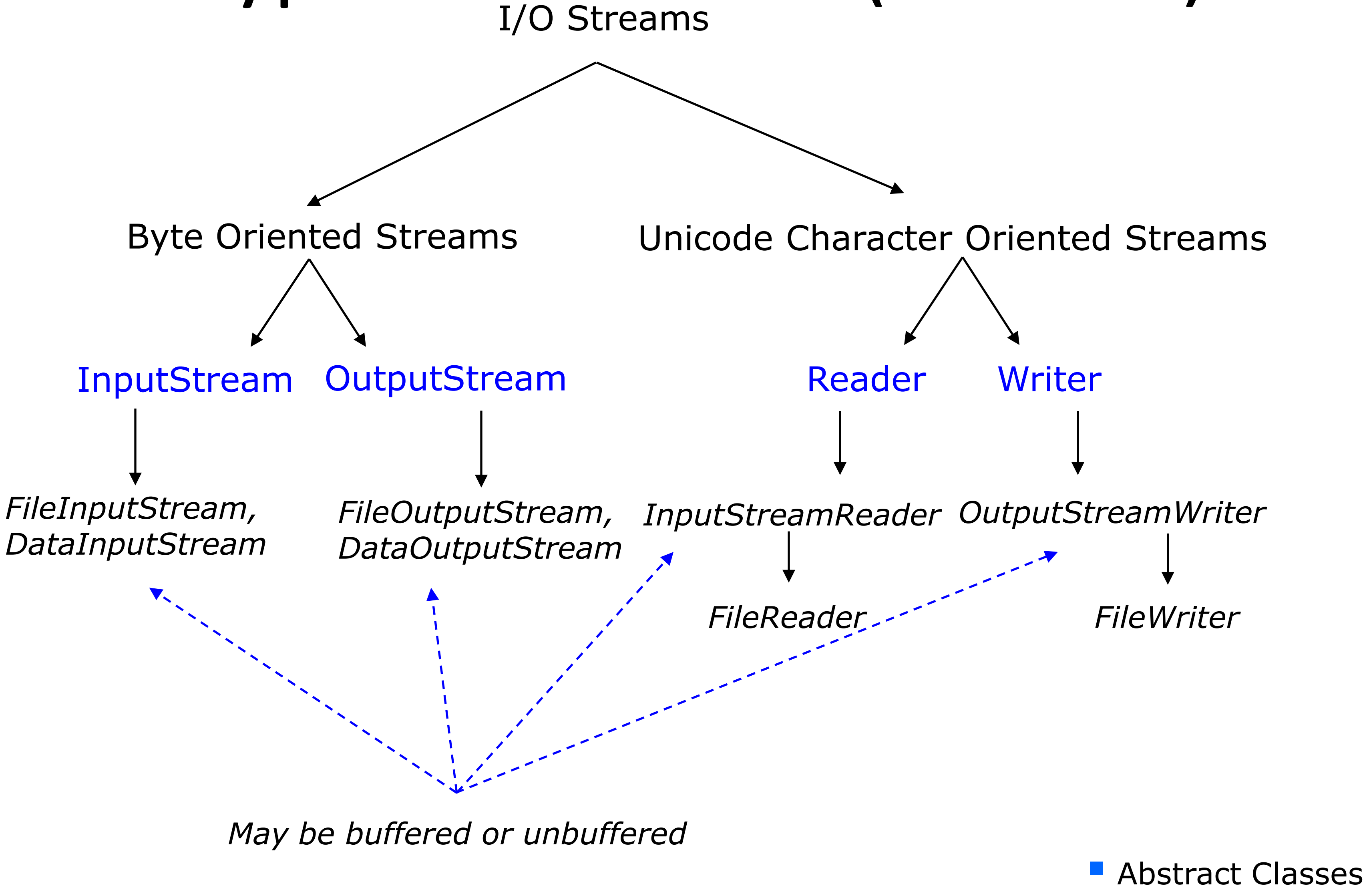
File.txt

Types of Streams

Two Types of Stream Classes:

- Byte Stream
  - Usually works for bytes & binary objects
  - *InputStream* and *OutputStream* are the abstract classes which represent Byte Streams
- Character Stream
  - Usually works for Characters & Strings
  - Follows the Unicode
  - *Reader* and *Writer* are the abstract classes which represents Character Streams

Types of Streams (Contd…)



I/O Streams

Byte Oriented Streams

Unicode Character Oriented Streams

InputStream

OutputStream

Reader

Writer

FileInputStream, DataInputStream

FileOutputStream, DataOutputStream

InputStreamReader

OutputStreamWriter

FileReader

FileWriter

May be buffered or unbuffered

🟦 Abstract Classes

Give this a Try…

1. Character Stream uses _______ standard to represent characters.

2. To write the data to the file which stream is to be used?

Byte Streams

- *FileOutputStream* & *FileInputStream* classes:
  - These are sub classes of *OutputStream* and *InputStream* classes respectively
  - Used to write & read binary data and /or binary object to and from the data source

```
FileOutputStream fos = new FileOutputStream(“abc.txt”);

FileInputStream fis = new FileInputStream(“abc.txt”);
```

Byte Streams (Contd…)

- *FileInputStream* object is used to read data from the file

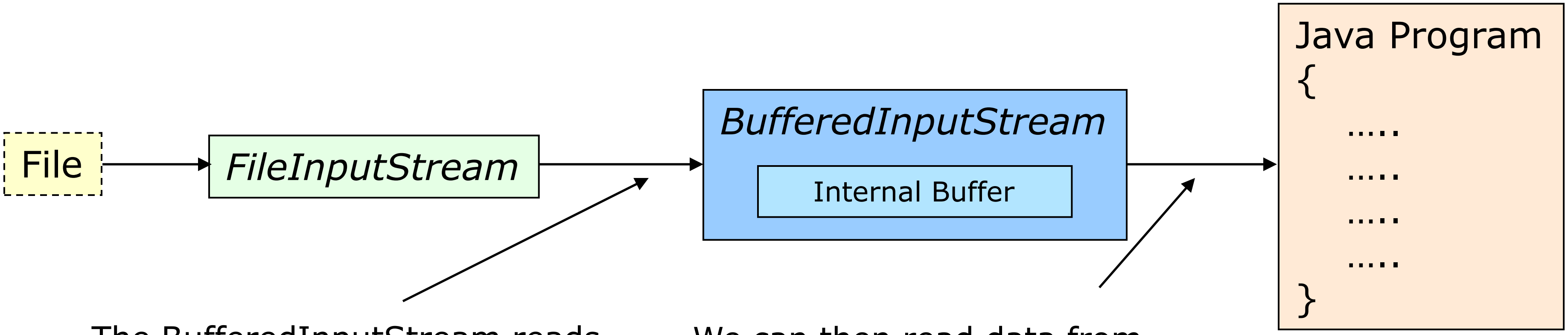
```
FileInputStream testFile;

try {
       testFile = new FileInputStream(“test.dat”);

        while((nextByte = testFile.read()) != -1)
       {
       System.out.println(nextByte);
       }
     }
    catch(IOException e)
    {
   System.out.println(“Error reading file + e ”);
    }
```

read() returns -1 on encountering end of file

ReadWriteFile.java (ReadWriteFile.java.JAV)

Byte Streams (Contd…)

- *BufferedInputStream* & *BufferedOuputStream* classes:
  - Subclasses of *InputStream* & *OutputStream* classes respectively
  - We can wrap a *BufferedInputStream* around the *FileInputStream* for reading & storing large chunks of data in a buffer at once for later use



The BufferedInputStream reads data from the File in large chunks and stores the data in an internal buffer

We can then read data from the *BufferedInputStream*. Data is read from the buffer instead of directly from the file on each read

Byte Streams (Contd…)

BufferedInputStream adds buffering to *FileInputStream* object

```
BufferedInputStream bufferedFile;

try {
      bufferedFile = new BufferedInputStream(new
      FileInputStream(“test.dat”);

       while((nextByte = bufferedFile.read()) != -1)
      {
            System.out.println(nextByte);
      }
    }
   catch(IOException e)
   {
  System.out.println(“Error reading file + e ”);
    }
```

BufferedTest.java

Give this a Try…

1. Buffer Stream is used to read large amount of data as compare to File Stream? State True / False

2. Can we append the data in the existing file?

3. If the file to be readm does not exists, which exception gets thrown?

Byte Streams (Contd…)

- Data I/O Streams

  - We may want an even higher level of abstraction and wish to read & write data to and from streams in the form of primitive data variables (rather than just bytes or characters)

  - Java has built in stream classes to automatically handle converting this information into the necessary raw bytes that a stream can use

Byte Streams (Contd…)

- *DataInputStream* & *DataOutputStream* classes:
  - Allow to read and write primitive data types to input and output streams respectively

```
BufferedOutputStream bufStream;
DataOutputStream dataStream;
try {
   bufStream = new BufferedOutputStream(new
         FileOutputStream(“file.out”);
   dataStream = new DataOutputStream(bufStream);
   dataStream.writeInt(5);
} catch(IOException e) {
   System.out.println(“Error writing to file “ + e );
} finally {
  // Write code in try/catch to close the streams
}
```

DataStream.java (DataStream.java.JAV)

Object Serialization

- The process of writing the state of an object to a byte stream
- Saves the state of an Object to any data destination like file
- This may later be restored by the process of Deserialization
- Only an object that implements the *Serilizable* interface can be saved & restored by the serilization facilities
- The *Serializable* interface defines no members; It is simply used to indicate that a class may be serialized
- All subclasses of a *serializable* class are also *serializable*
- transient declared & static variables are not saved by this

Object Serialization (Contd…)

- *ObjectOutputStream* & *ObjectInputStream* classes
  - Subclasses of *InputStream & OutputStream* classes
  - Same functionality as *DataInputStream* & *DataOutputStream*
  - Also include support for reading and writing objects data via the *readObject()* & *writeObject()* methods

```
ObjectOutputStream oos = new ObjectOutputStream(
new FileOutputStream(“abc.txt”));
oos.writeObject();

ObjectInputStream ois = new ObjectInputStream(
new FileInputStream(“abc.txt”));
ois.readObject();111
```

Serial.java (Serial.java.JAV)

Give this a Try…

1. To read the specific primitive data type which Stream class we have to use?

2. Which interface is used to control the Serialization?

Character Streams

- Two types of Character Stream classes:

  1. Reader
  2. Writer

FileReader and *FileWriter* classes:

- Subclasses of *Reader* & *Writer* class
- Used to read and write characters or strings from a data source like file

```
FileWriter fw = new FileWriter(“abc.txt”,true);

FileReader fr = new FileReader(“abc.txt”) throws
IOException
```

Character Streams (Contd…)

- *BufferedReader* & *BufferedWriter* classes:
  - Provides buffering to Character streams
  - Subclasses of *Reader* & *Writer* classes
  - *BufferedReader* is used to read data from console & files
- *InputStreamReader* class:
  - Serves as a wrapper for any *InputStream* object
  - Converts the raw bytes as they are read from the *InputStream* and serves them to the user as Unicode characters

Character Streams (Contd…)

- Reading data from console:
  - We can wrap *InputStreamReaders* around *InputStreams* to make them useful in reading character data
  - *BufferedReader* provides a *readLine()* method for additional functionality

```
BufferedReader stdin = new BufferedReader(new
        InputStreamReader(System.in));
try {
   String input = stdin.readLine();
   System.out.println("The input from the command line
                    was " + input);
} catch(IOException e) {
   System.err.println("Error reading data");
}
```

BRReadLines.java

Give this a Try…

1. Which method is used to read the data line by line from the console?

2. Reader class deals with which kind of encoding?

Core Java

Introducing GUI Programming

AWT & Swings

- Abstract Windowing Toolkit (AWT): is the set of components used to design the GUI
- These components have platform specific code
- Swing components:
  - Advance features like setting Border to the component
  - Do not have any platform specific code

FrameAppln.java

Tab.java

Introduction to Applets

- A Java program that runs inside a browser
  - We embed applet in an HTML page using the <APPLET> tag
  - Introduces interactivity in static web pages
  - Applets run inside a browser (executed at client side) & save server roundtrip, thereby reducing traffic
- *Appletviewer* is a test utility available in JDK for testing applets if we do not have a java-enabled browser
- <applet> tag has attributes like:
  - Code : Represents the .class file of the Applet
  - Height : Represents the height of the Applet
  - Width : Represents the width of the Applet

SmileApplet.java

Give this a Try…

1. What is the advantage of Swing Components over AWT Components?

2. Which utility is used to execute the Applet Program on the console?

3. Who will handle the event which is fired by any Component?

Summary

In this session, we have covered:

- What is I/O Stream
- Types of Streams
- Byte Streams
- Object Serialization
- Object Externalization
- Character Streams
- Introduction to AWT & Swings
- Introduction to Applets

Thank You