Basic Java : Streams & Files

Persistent University

PERSISTENT

Objectives :

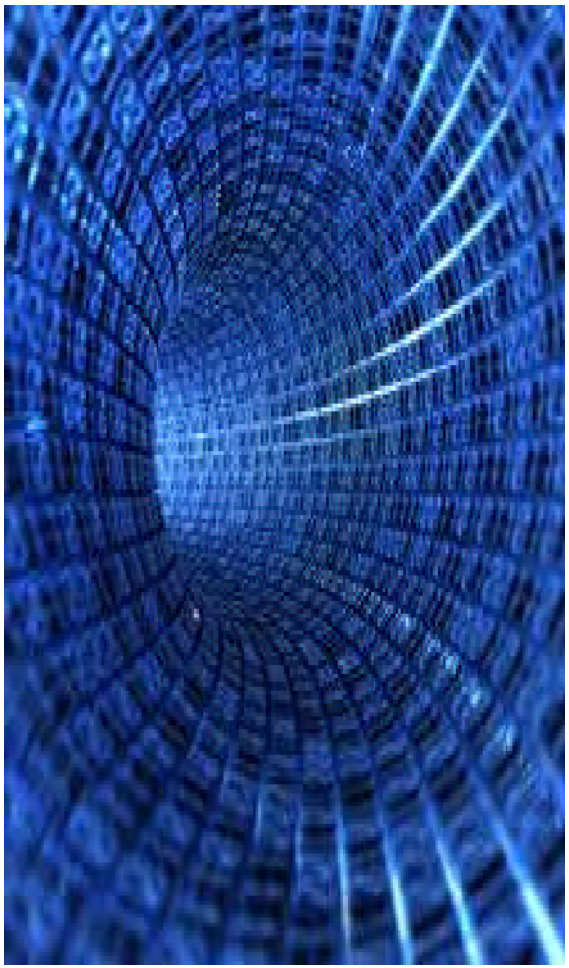
- At the end of this module, you will be able to understand :
  - Files and Streams
  - File Handling and File IO
  - csv, txt, property file reading
  - JDK 1.6 : Console class

Input - Output

- Need
  - Building interactive systems
  - Persistent data storage
  - Convenient interaction with various type of devices. Complete abstraction from the device's behavior.
  - Handling large volumes of data
  - Working with data mediums
    - Characters, Bytes, Objects, Beans, XML etc.

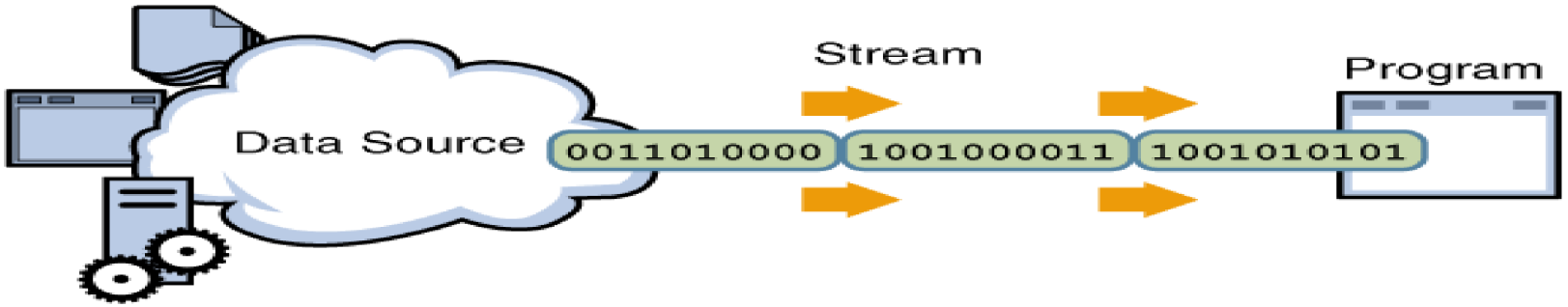
Streams

- What is a stream ?
  - Ordered sequence of data
  - A channel of data flow
  - An interface between your program and I/O devices

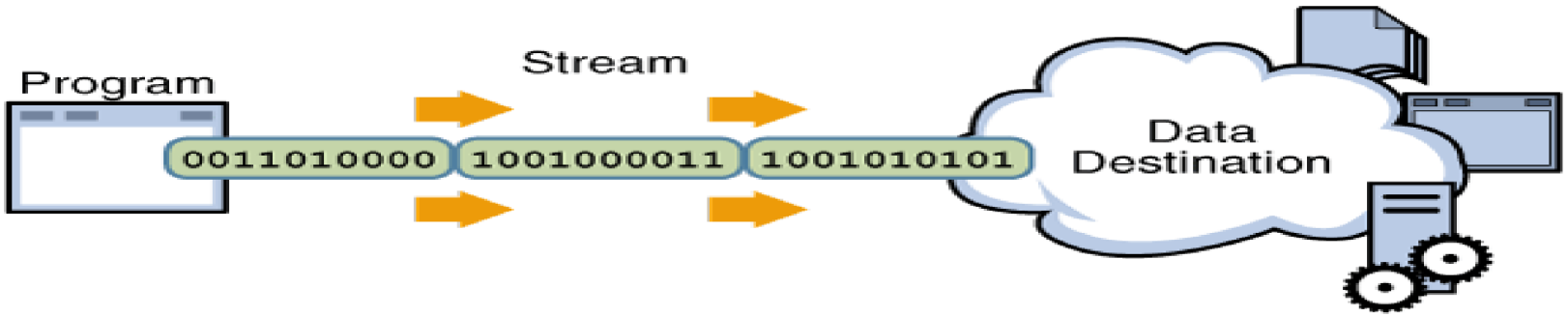


Types of streams

- **Input**

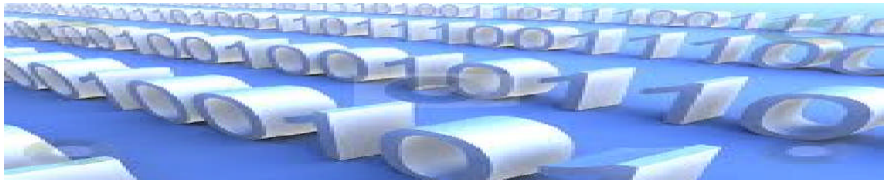


- **Output**



Java I/O stream categories

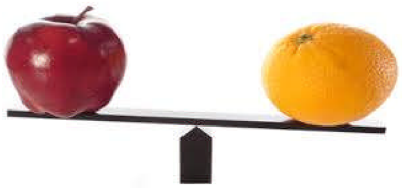
- Byte / Binary
  - 8-bit data unit



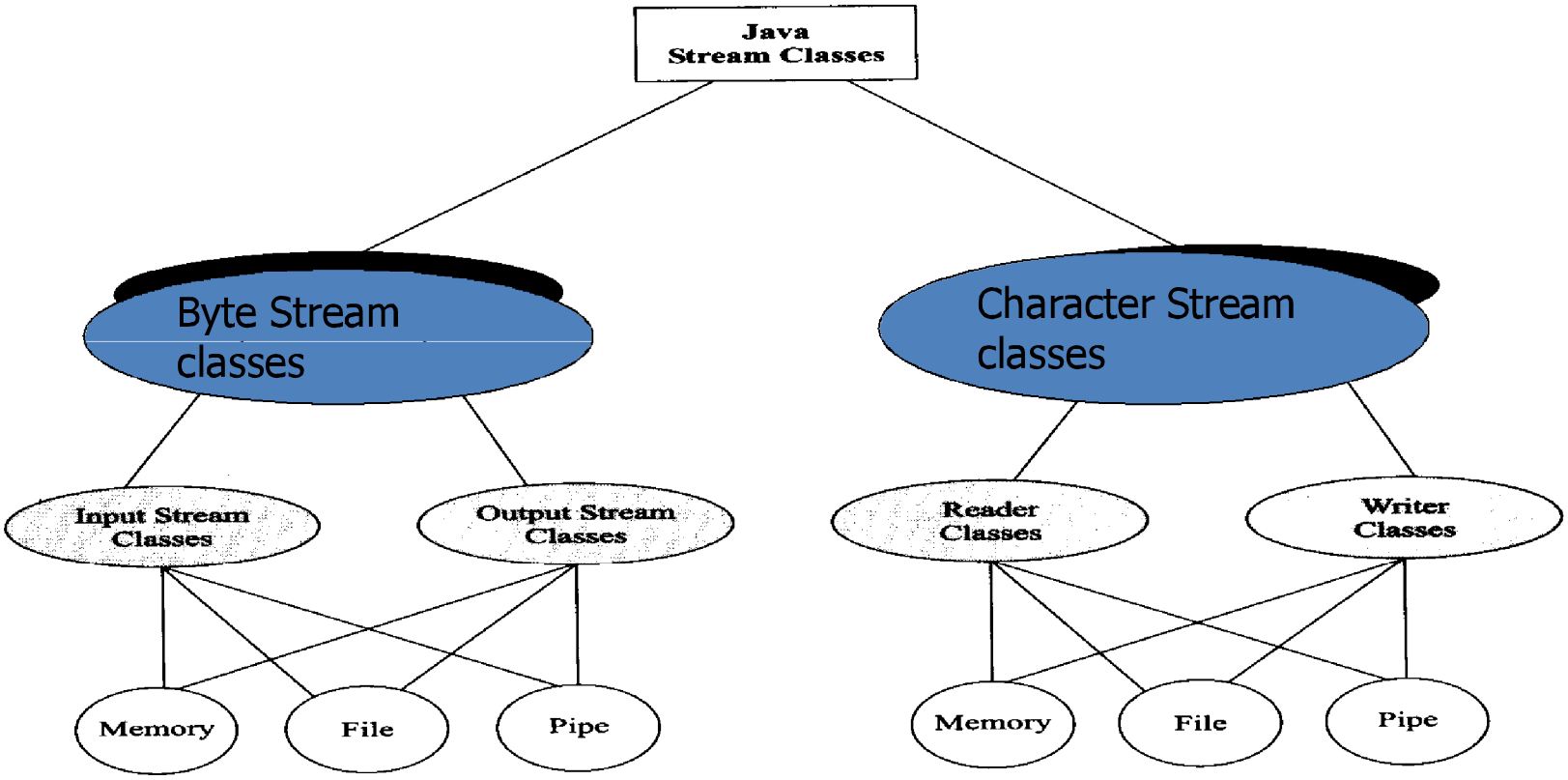
- Character / Text
  - 16-bit data unit
  - Support for internationalization through Unicode

Byte vs. Character Streams

Byte Streams	Character streams
Operated on 8 bit (1 byte) data.	Operates on 16-bit (2 byte) unicode characters.
Input streams/Output streams	Readers/ Writers

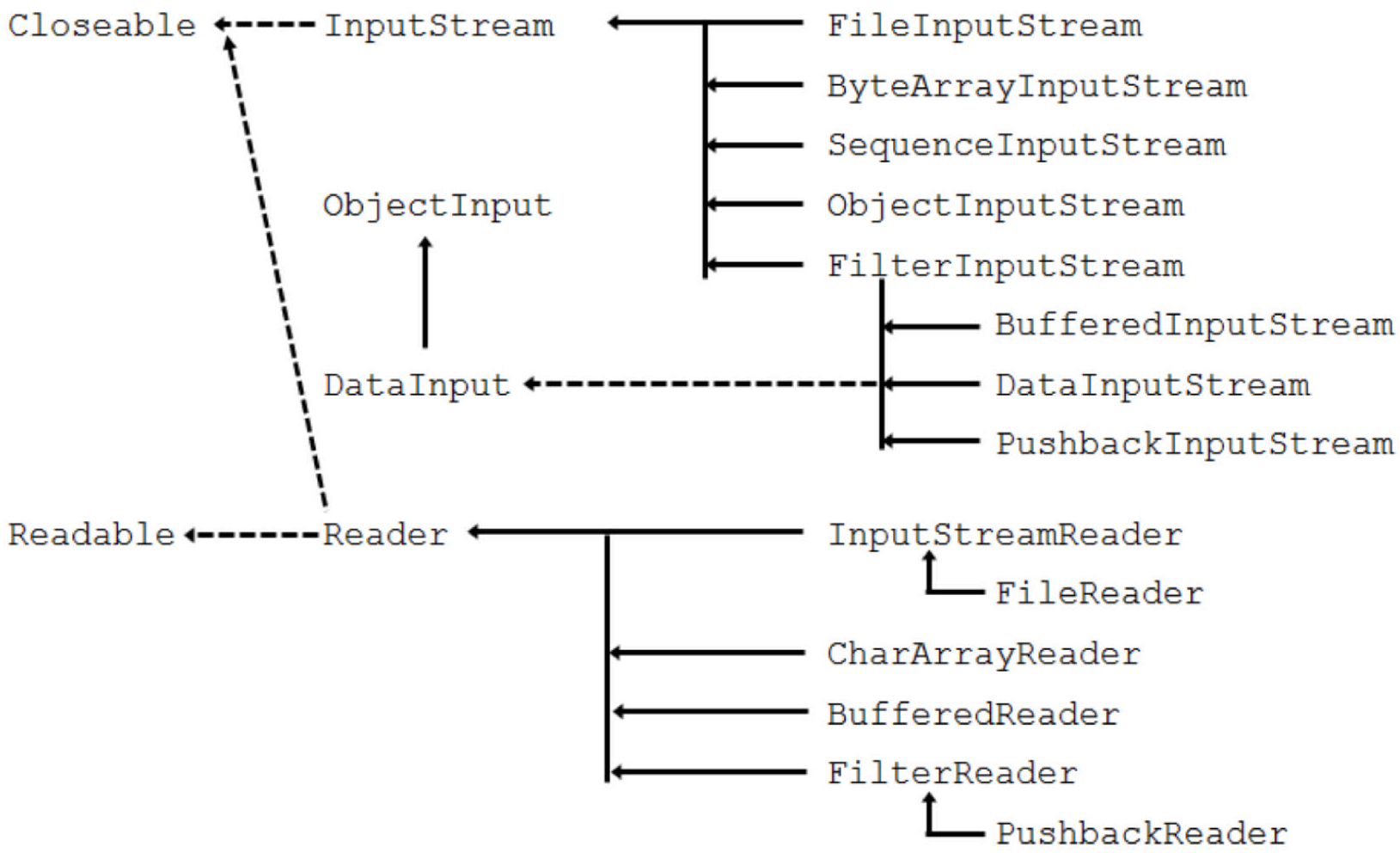


Classification of Java Stream Classes

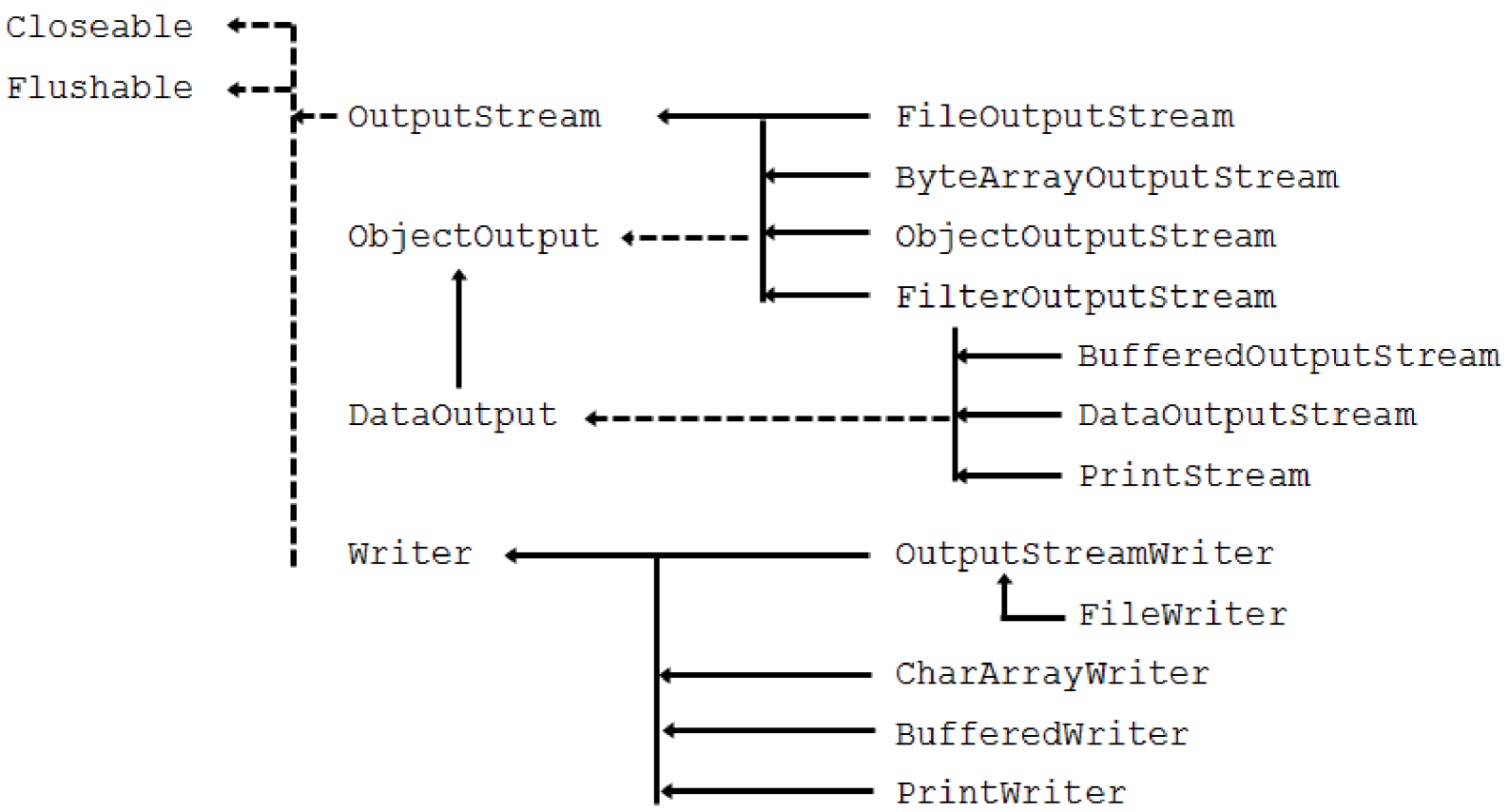


Classification of Java stream classes

The I/O class hierarchy : Input



The I/O class hierarchy : Output



Java I/O – Using OutputStreams

- Basic pattern for output is as follows:
  - Open a stream
  - While there's data to write
  - Write the data
  - Close the stream

Byte streams - OutputStreams

```
/* construct a file output stream for writing bytes to a disk file */
FileOutputStream fileOutputStream1;
fileOutputStream1 = new FileOutputStream("image.jpg");

/* construct a file output stream in append mode for writing bytes to a disk file
   */
FileOutputStream fileOutputStream2;
fileOutputStream2 = new FileOutputStream ("program.exe", true);

/* construct a file output stream with a java.io.File object for writing bytes to a
   disk file */
FileOutputStream fileOutputStream3;
fileOutputStream3 = new FileOutputStream (new File("lib.so"));
```

OutputStreams – Methods and exceptions

- Key methods

void write(int)	Writes a single byte. The 8 low-order bits are written.
void write(byte [])	Writes an array of bytes
void close()	Close the stream.

- Key exception
  - `java.io.IOException`
  - `java.io.FileNotFoundException`

Java I/O – Using InputStreams

- Basic pattern for I/O programming is as follows
  - Open a stream
  - While there's data to read
  - Process the data
  - Close the stream

Byte streams - InputStreams

```
/* construct a file input stream for reading bytes from a disk file */
FileInputStream fileInputStream1;
fileInputStream1 = new FileInputStream ("image.jpg");


/* construct a file input stream with a java.io.File object for reading bytes from a
     disk file */
FileInputStream fileInputStream2;
fileInputStream2 = new FileInputStream (new File("lib.so"));
```

InputStreams – Methods and exceptions

- Key methods

`int read()`	Read and return a single byte or -1 if end of the stream is reached.
`int read(byte [])`	Read an array.length of bytes into the array. Return the number of bytes read or -1.
`long skip(long)`	Skip past some bytes in the stream. Returns the number of bytes actually skipped.
`void close()`	Close the stream.

- Key exceptions
  - `java.io.FileNotFoundException`
  - `java.io.IOException`

.properties file reading

- .properties files are mainly used in Java related technologies to store the configurable parameters of an application.
- The parameters are stored in properties file in the form of key-value pair.
- Key specifies name of the property and value indicates the value for that property.

Steps to read properties file

- Create .properties file
- Load the file
- Read properties using **getProperty()** method.

Demo : Reading properties file

- Consider following properties file named as data.properties

data.properties

```
1 property1=value1
2 property2=value2
3 property3=value3
4 property4=value4
```

Demo : Reading properties file

```
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;
public class ReadPropertiesFile {
    public void readFile(){
        Properties properties = new Properties();
        InputStream inputStream = null;
        try {
            inputStream = new FileInputStream("data.properties");
            properties.load(inputStream);
            String val1 = properties.getProperty("property1");
            String val2 = properties.getProperty("property2");
```

Demo : Reading properties file

```
String val3 = properties.getProperty("property3");
String val4 = properties.getProperty("property4");

System.out.println("Value 1 : " + val1);
System.out.println("Value 2 : " + val2);
System.out.println("Value 3 : " + val3);
System.out.println("Value 4 : " + val4);
} catch (IOException e) { e.printStackTrace();}  }   }
public static void main(String[] args) {
        ReadPropertiesFile propertiesFile = new ReadPropertiesFile();
        propertiesFile.readFile();  }    }
```

Console class

- Added in java.io package.
- Provides the methods to access the character-based console device, associated with the current JVM.
- Refers the console associated with operating system.

Console class

- Provides facility to read text & password.
- Reads the password from the console without displaying it to the user.
- **readPassword()** method provides this facility.

How to get object of Console class?

- A static method , console() of System class returns object of Console class

```
Console console = System.console();
```

How to use Console class

```
import java.io.Console;
public class ConsoleDemo {
  public static void main(String[] args) {
            Console console = System.console();
            String name = console.readLine("Enter name: ");
            System.out.println(name);
            char ch[] = console.readPassword("Enter password: ");
            System.out.println(ch);
    }
}
```

To remember while using Console class

- Console class represents the operating system console and not console of eclipse.
- Make a note not to use Console class in eclipse else NullPointerException will be thrown.
- Execute the program using command prompt.

Summary : Session

With this we have come to an end of our session, where we discussed about ....

- Streams and files
- How to use streams
- Console class

At the end of this session, we see that you are now able to answer following questions:

- What are streams?
- What are types of streams?
- How to read & write streams?
- How to read .properties file?

Reference Material : Websites & Blogs

- http://www.javatpoint.com/FileInputStream-and-FileOutputStream
- http://www.javatpoint.com/FileReader-and-FileWriter
- http://www.tutorialspoint.com/java/java_files_io.htm

Reference Material : Books

- ***Head First Java***
  - *By: Kathy Sierra, Bert Bates*
  - *Publisher: O'Reilly Media, Inc.*

- ***Java Complete Reference***
  - *By Herbert Schildt*

Key Contacts :

Java Interactive :

- Asif Immanad

asif_immanad@persistent.co.in

- Nisha Waikar

nisha_waikar@persistent.co.in

- Varsha Mulik

varsha_mulik@persistent.co.in

Persistent University :

- Poorva Kulkarni

poorva_kulkarni@persistent.co.in

Vice President :

- Shubhangi Kelkar

shubhangi_kelkar@persistent.co.in

Thank You !!!

Persistent University