COCSIT

# IO STREAMS INTRODICTION

COCSIT


# I/O Streams- Introduction

# Agenda

1. Introduction to I/O Streams
2. Predefined I/O Streams

# Objectives

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

- Understand I/O Streams and its categories
- Understand about predefined I/O Streams



# I/O Streams




# I/O Streams

- Java programs perform I/O through streams. A stream is:
  - an abstraction that either produces or consumes information
  - linked to a physical device by the Java I/O system
- All streams behave similarly, even if the actual physical devices to which they are linked differ.
- Thus the same I/O classes can be applied to any kind of device as they abstract the difference between different I/O devices.

# I/O Streams (Contd.).

- Java's stream classes are defined in the **java.io** package.
- Java 2 defines two types of streams:
  - **byte streams**
  - **character streams**
- Byte streams:
  - provide a convenient means for handling input and output of bytes
  - are used for reading or writing binary data
- Character streams:
  - provide a convenient means for handling input and output of characters
  - use **Unicode**, and, therefore, can be internationalized

# The Predefined Streams

- System class of the java.lang package contains three predefined stream variables, in, out and err.
- These variables are declared as public and static within System:
  - System.out refers to the standard output stream which is the console.
  - System.in refers to standard input, which is the keyboard by default.
  - System.err refers to the standard error stream, which also is the console by default.

# Difference between System.out and System.err

- System.out sends the output to the standard output stream, which is console by default.
- System.err sends the output to the standard error stream, which also happens to be console by default
- The reason behind having two separate streams for output and error is that the standard o utput should be used for regular program outputs while standard error should be used for error messages.

# System.out and System.err (Contd.).

- Both these streams can be redirected to different destinations.
- We can redirect the program output to a particular log file and the error messages to another log file by using the following syntax :

```
java StreamDemo > output.log 2>error.log
```

(where StreamDemo is the name of the class)

During execution, the program output will be stored in output.log while the error message(if any) will be stored in error.log

# Demonstration of System.out and System.err

```
class StreamDemo {
    public static void main(String[] args) {
        try {
    System.out.print("Writing program output to the output file ");
    int i=0;
    int z=100/i;
        }
        catch(Exception e) {
    System.err.print("ArithmeticException has occured");
        }
    }
}
```

# System Properties

- System Properties provide information about local system configuration.
- When the Java Virtual Machine starts, it inserts local System Properties into a System properties list.

- We can use methods defined in System class to access or change the values of these properties.

```
public static Properties getProperties()
public static String getProperty(String key)

public static void setProperties(Properties prp)
```

# System Properties(contd.).

Some of the Important Properties are listed below :

| Key | Description of Associated Value |
|---|---|
| java.version | Java Runtime Environment version |
| java.home | Java installation directory |
| java.class.path | Java class path |
| os.name | Operating system name |
| user.name | User's account name |
| user.home | User's home directory |
| user.dir | User's current working directory |

# System.getProperties()

public static Properties getProperties()

The **System.getProperties()** method returns an object of the type Properties.

You can use this method to list all the System Properties.

# Demo of System.getProperties() method

```
import java.util.*;
class GetPropertiesDemo {
    public static void main(String [] args) {
        Properties x = System.getProperties();
        x.list(System.out);
    }
}
```

# System.getProperty (String Key)

public static String getProperty(String key)

You can also use System.getProperty(String Key) method to get the value of a particular property represented with a key.

# System.getProperty (String Key) - Demo

```
class GetPropertyDemo {
    public static void main(String[] args) {
        String user_home= System.getProperty("user.home");
        String java_version = System.getProperty("java.version");
        String java_home = System.getProperty("java.home");
        String class_path = System.getProperty("java.class.path");
        String os_name = System.getProperty("os.name");
        String user_name = System.getProperty("user.name");
        String user_dir = System.getProperty("user.dir");
        System.out.println("The user home directory is "+user_home);
        System.out.println("The java version is "+java_version);
        System.out.println("The Java Home directory is "+java_home);
        System.out.println("The class path is set to "+class_path);
        System.out.println("The Operating System is "+os_name);
        System.out.println("The user name is "+user_name);
        System.out.println("The working directory is "+user_dir);
    }
}
```

# System.getProperty (String Key) - Demo

```
class GetPropertyDemo {
    public static void main(String[] args) {
        String user_home= System.getProperty("user.home");
        String java_version = System.getProperty("java.version");
        String java_home = System.getProperty("java.home");
        String class_path = System.getProperty("java.class.path");
        String os_name = System.getProperty("os.name");
        String user_name = System.getProperty("user.name");
        String user_dir = System.getProperty("user.dir");
        System.out.println("The user home directory is "+user_home);
        System.out.println("The java version is "+java_version);
        System.out.println("The Java Home directory is "+java_home);
        System.out.println("The class path is set to "+class_path);
        System.out.println("The Operating System is "+os_name);
        System.out.println("The user name is "+user_name);
        System.out.println("The working directory is "+user_dir);
    }
}
```

# System.setProperties()

public static void setProperties(Properties prp)

The **System.setProperties(Properties prp)** method sets the system properties to the Properti es argument.

You can use this method to change the system properties as per your requirement.

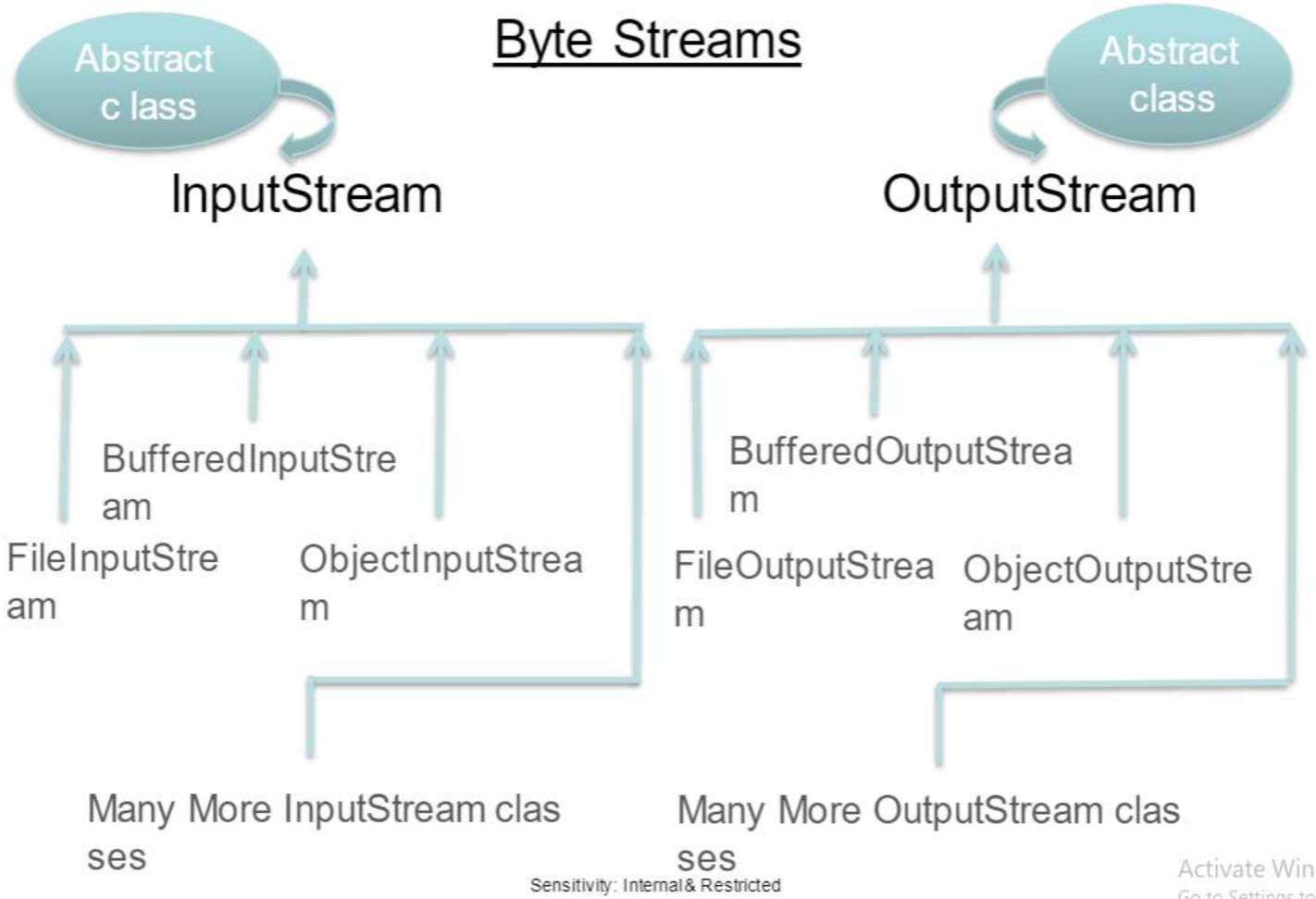
# System.setProperties() - Demo

```
import java.util.*;
public class SystemPropertiesDemo {
   public static void main(String[] args) {
     System.out.print("Previous value of Java Home : ");
     System.out.println(System.getProperty("java.home"));

     Properties p = System.getProperties();
     p.put("java.home", "D:\\Program Files\\java1.5\\jdk1.5.0_02");
     System.setProperties(p);

     System.out.print("New value of Java Home : ");
     System.out.println(System.getProperty("java.home"));
   }
}
```

*The above code when executed, prints :*

**Previous value of Java Home : D:\Program Files\java1.6\jdk1.6.0_05\jre**
**New value of Java Home : D:\Program Files\java1.5\jdk1.5.0_02**

# I/O Streams hierarchy



# Byte Stream classes

**Buffered**InputStream
**Buffered**OutputStrea
m

To read & write data into buffer

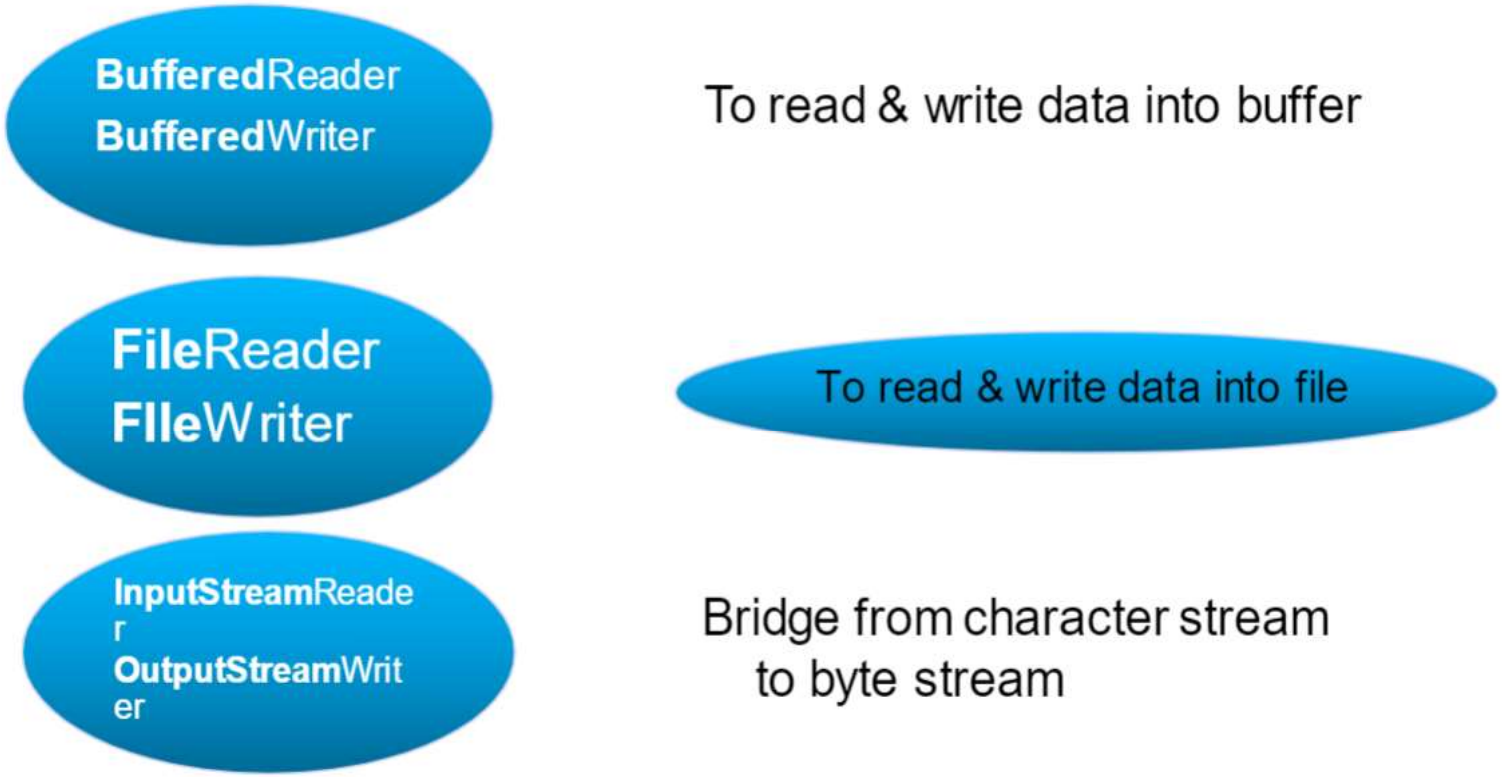
**File**InputStream
**FIle**OutputStream

To read & write data into file

**Object**InputStream
**Object**OutputStrea
m

To read & write object into secondary device (serialization )

# Character Stream classes



# Match the following

- Match the streams with the appropriate phrases in column B

| Column A | Column B |
|---|---|
| 1. FileWriter | Byte stream for reading from file |
| 2. FileInputStream | Character stream for reading from file |
| 3. FileOutputStream | Character stream for writing to a file |
| 4. FileReader | Byte stream for writing to a file |

# Quiz

1. Java input output classes are available in _____package
   a. java.lang
   b. java.io
   c. java.util

2. What are the inbuilt streams available in java.io package
   a. System.in
   b. System.out
   c. System.err
   d. All of the above
3. Can data flow through a given stream in both directions?
   a. Yes
   b. No

# Summary

- Introduction to I/O Streams
- Predefined I/O Streams


# Thank You