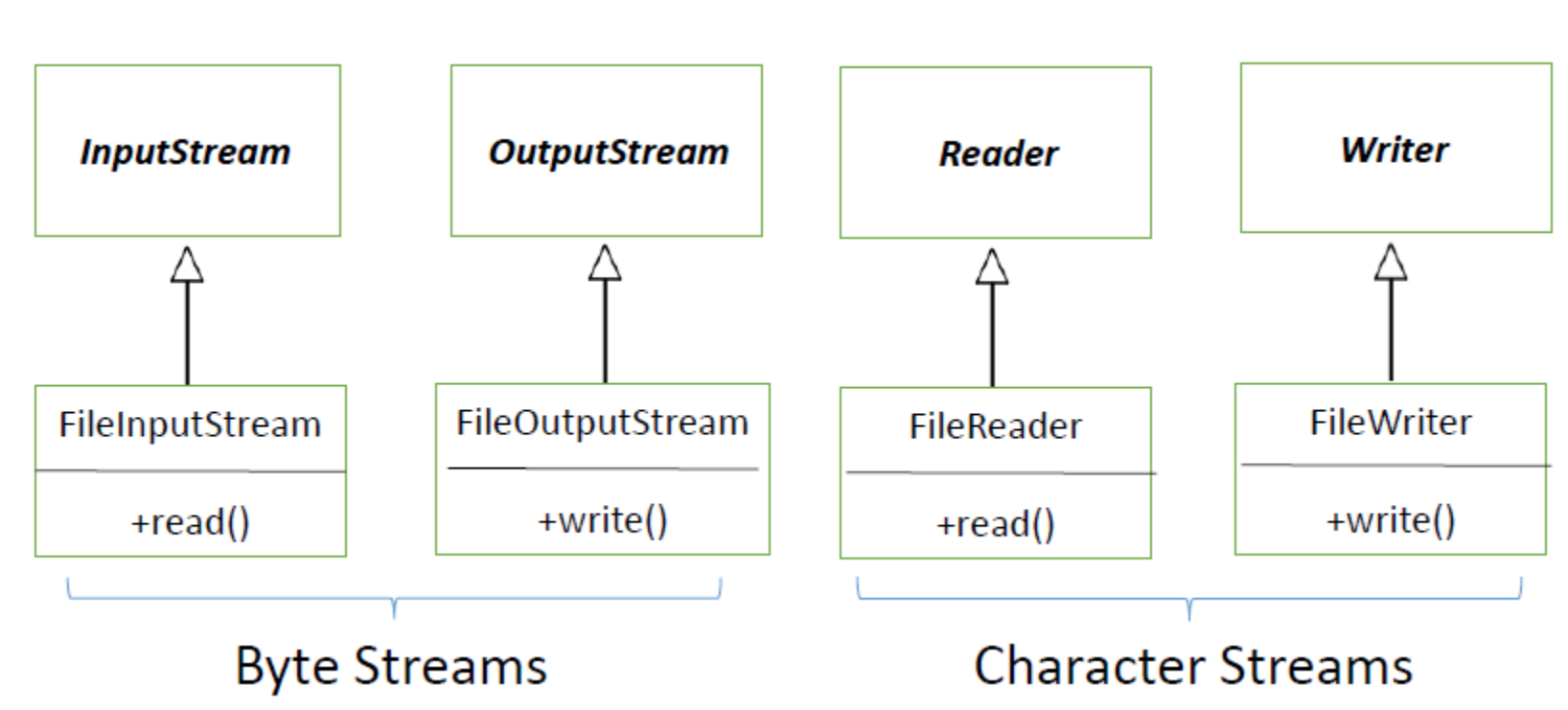
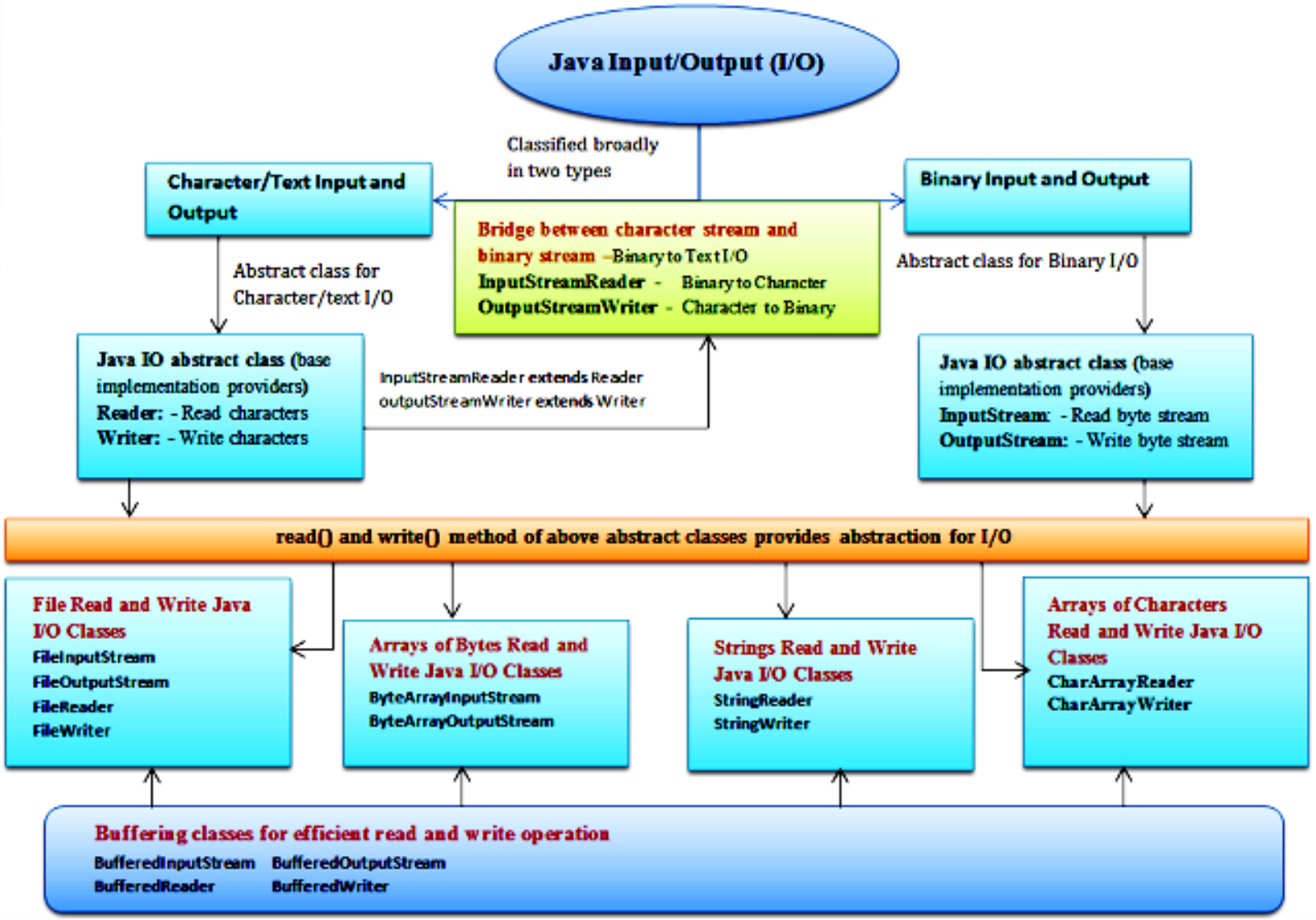
Lecture 22 contd…

Prof. Anita Agrawal
BITS-Pilani, K.K.Birla Goa Campus

Java I/O Fie Handling



Following diagram depicts a holistic view of Java I/O and classes involved to support both text I/O and binary I/O.



Different ways of reading and writing a text file

- All programming languages provide support for standard I/O where the user's program can take input from a keyboard and then produce an output on the computer screen.

- Java provides the following three standard streams:
- **Standard Input** − This is used to feed the data to user's program and usually a keyboard is used as standard input stream and represented as **System.in** (object of inputstream)
- **Standard Output** − This is used to output the data produced by the user's program and usually a computer screen is used for standard output stream and represented as **System.out**. **(**object of printstream)
- **Standard Error** − This is used to output the error data produced by the user's program and usually a computer screen is used for standard error stream and represented as **System.err**. **(**object of printstream)

System class

- All Java programs automatically import the **java.lang** package
- This package defines a class called **System**
- System class is final and all of it's members and methods are **static** so that we can't subclass and override it's behavior through **inheritance**.
- System class in java doesn't provide any public constructors. So we can't instantiate this class and hence all of it's methods are static

- **System** also contains three predefined stream variables
  - **in**, **out**, and **err**
    - System.in
    - System.out
    - System.err

Reading and writing a file

- Reading a file:
  - File Reader class
  - Buffered Reader class
  - Scanner Class

Illustrating methods

.available()

.read(destination, offset, length)

Scanner class

- Java has various ways to read input from the keyboard, the Scanner class is one of them.
- The scanner class is found in the java.util package.
- **Scanner** can be used to read input from the console, a file, and a string.
- Scanner class is used for obtaining the input of the primitive types like int, double etc. and strings.

- To read from the standard input (keyboard), create an object of Scanner class, by passing the predefined object System.in, which represents the standard input stream.

```
Scanner S1 = new Scanner(System.in);  //scanner1.java
```

- To read from the file, Read1.txt, create an object of Scanner class, as follows:

```
FileReader fr = new FileReader("Read1.txt");  //scanner2.jav
Scanner s1= new Scanner(fr);
                or
File file = new File("Read1.txt");
Scanner s1 = new Scanner(file);
                or
```

```
FileInputStream fr = new FileInputStream("Read1.txt");
Scanner s1 = new Scanner (fr);
```

- To read the string “This is an OOP class”, create an object of the scanner class as follows:

```
String str  = “This is an OOP class”     //scanner3.java
Scanner S1 = new Scanner(str);
```

- Java Scanner reads tokens from the underlying source which could be a File or InputStream or a String. The tokens are usually separated by spaces.

- If we don't want to read all the tokens, but only want certain type of tokens we can do that by using the hasNext methods. // Scanner4.java

boolean hasNext()
boolean hasNextBoolean()
boolean hasNextByte()
boolean hasNextByte(int radix)
boolean hasNextDouble()
boolean hasNextFloat()
boolean hasNextInt()
boolean hasNextInt(int radix)
boolean hasNextLine()
boolean hasNextLong()
boolean hasNextLong(int radix)
boolean hasNextShort()
boolean hasNextShort(int radix)

1	**boolean**	**hasNextBoolean()**	**It is used to check if the next token in this scanner's input can be interpreted as a Boolean or not.**
2	**boolean**	**hasNextByte()**	**It is used to check if the next token in this scanner's input can be interpreted as a Byte or not.**
3	**boolean**	**hasNextDouble()**	**It is used to check if the next token in this scanner's input can be interpreted as a Double or not.**
3	**boolean**	**hasNextFloat()**	**It is used to check if the next token in this scanner's input can be interpreted as a Float or not.**
4	**boolean**	**hasNextInt()**	**It is used to check if the next token in this scanner's input can be interpreted as an int or not.**
5	**boolean**	**hasNextLine()**	**It is used to check if there is another line in the input of this scanner or not.**
6	**boolean**	**hasNextLong()**	**It is used to check if the next token in this scanner's input can be interpreted as a Long or not.**

The following steps are required in every Scanning operation:

- Determine if a specific type of input is available, by calling one of Scanner's hasNextXXXX methods.
- If input is available, read it by calling one of Scanner's nextX methods.
- Repeat the process until input is exhausted.
- Close the Scanner by calling close().

String next()
boolean nextBoolean()
byte nextByte()
byte nextByte(int radix)
double nextDouble()
float nextFloat()
int nextInt()
int nextInt(int radix)
String nextLine()
long nextLong()
long nextLong(int radix)
short nextShort()
short nextShort(int radix)

nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user
nextLine()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user

- The Console Class allows you to **read** from and **write** to the **console**
- Most of the functionality of the **Console** class is available through**System.in** and **System.out**
- Primarily a convenience class
  - Simplifying some type of console interactions such as reading strings from the console
- Console class doesn’t provide a constructor

- Obtain an object of Console class by calling the **console()** in **System** class
  - static Console console()

- Methods defined in **Console** class
  - readLine()
  - printf()
  - readPassword()