Java Programming

TEK RAJ CHHETRI

Course Objective

- Explain the Java programming environment
- Describe the concepts of programming elements using Java and object-oriented
- programming concepts
- Apply the exception handling and input/output in Java programming
- Apply the event handling, GUI programming using swing, and Java database connectivity

Unit 6: Input / Output

- Input/output Basics
- Console Input and Output
- Reading and Writing Files

Learning Outcome (Unit 6)

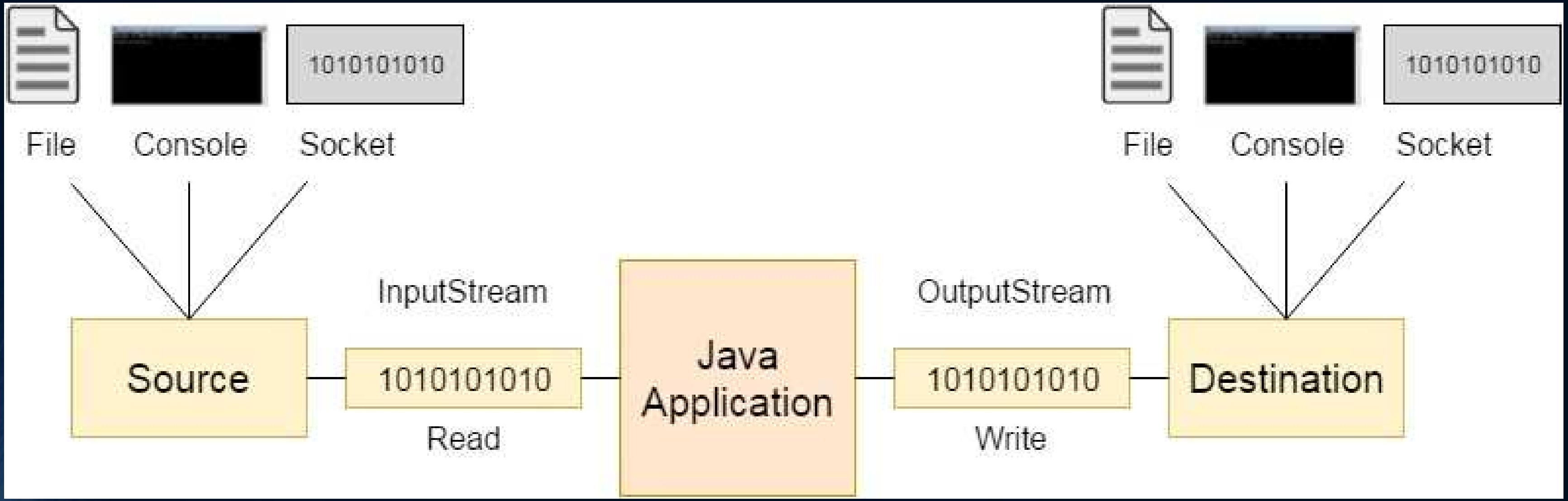
- Develop understanding about IO
- Able to read user input from console and write
- Able to write and write files

I/O Basics

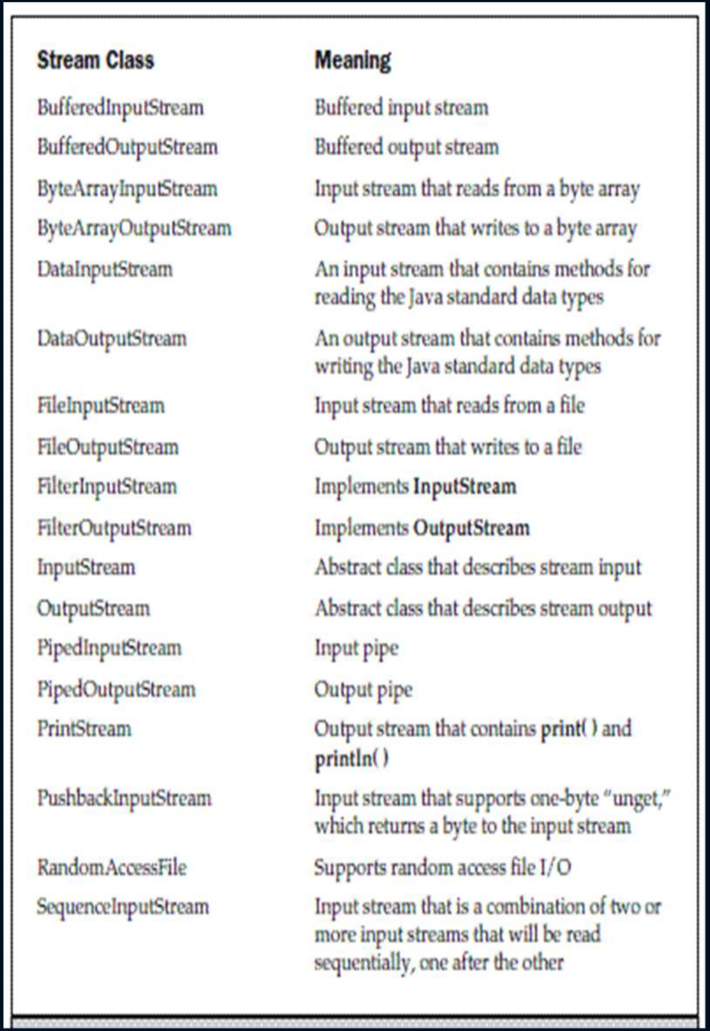
- Java I/O is used to process input and generate output.
- Separate IO package java.io for IO processing.
- Make use of streams for faster processing.

- Eg:
  - Reading user input from console, displaying the result, reading and writing content to file.

- Stream is an abstraction that produce or consume information.
- It is continuous data.
- All streams behave in the same manner, even with different physical devices.
- Three predefined stream in, out and err.
- Stream are of two types:
  1. Byte Streams: for handling input and output of bytes. Used for handling binary data.
  2. Character Streams: for handling I/O of characters. They use Unicode, hence can be internationalized.

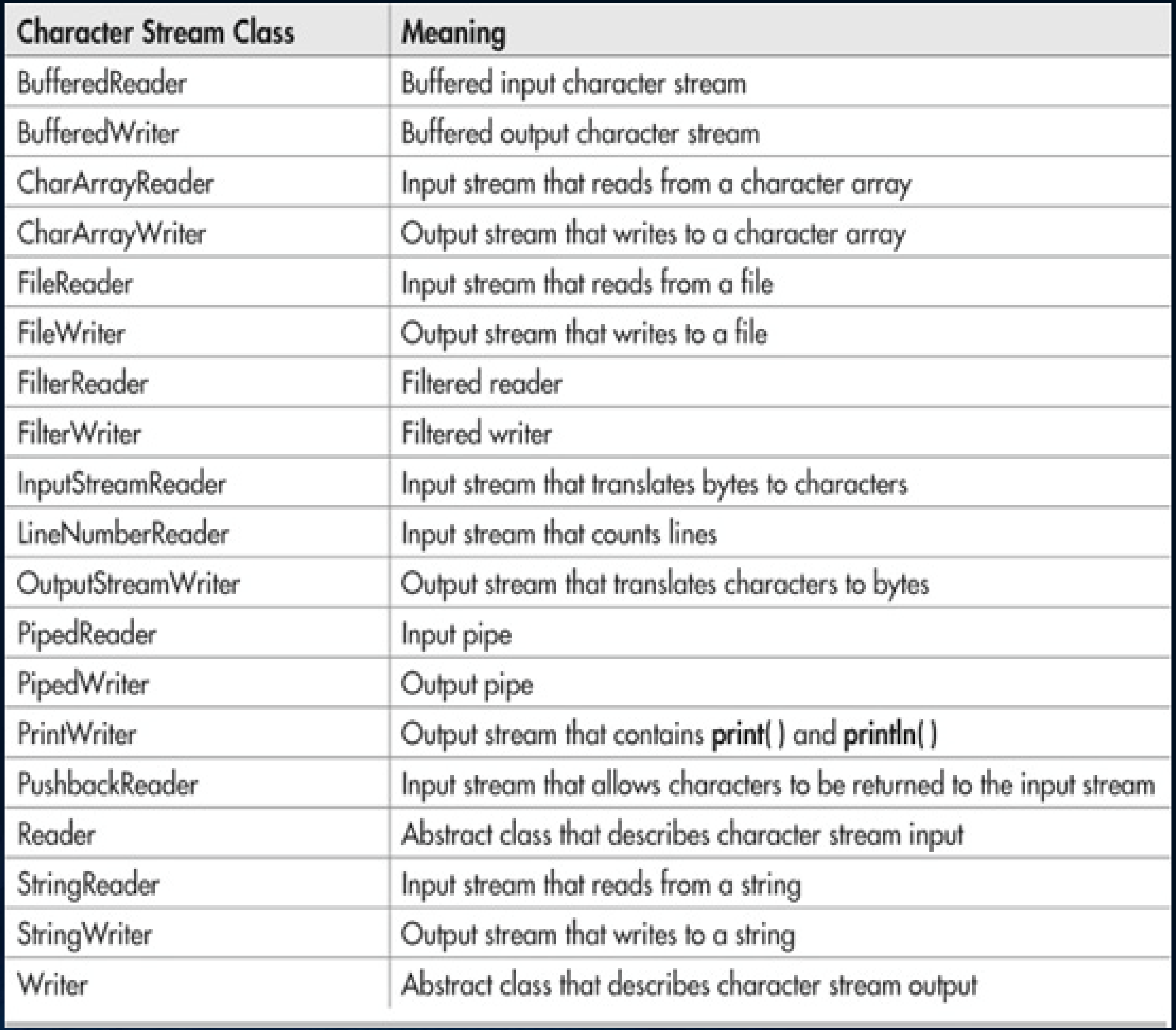

1010101010
File
Console
Socket
File
Console
Socket
1010101010
InputStream
OutputStream
Source
1010101010
Java Application
1010101010
Destination
Read
Write

- Byte streams and character streams are defined using two class hierarchies.
- At the top are two abstract classes: **InputStream** and **OutputStream** for byte stream class and **Reader** and **Writer** for character stream class.
- Two important methods are read() and write()



Stream Class	Meaning
BufferedInputStream	Buffered input stream
BufferedOutputStream	Buffered output stream
ByteArrayInputStream	Input stream that reads from a byte array
ByteArrayOutputStream	Output stream that writes to a byte array
DataInputStream	An input stream that contains methods for reading the Java standard data types
DataOutputStream	An output stream that contains methods for writing the Java standard data types
FileInputStream	Input stream that reads from a file
FileOutputStream	Output stream that writes to a file
FilterInputStream	Implements **InputStream**
FilterOutputStream	Implements **OutputStream**
InputStream	Abstract class that describes stream input
OutputStream	Abstract class that describes stream output
PipedInputStream	Input pipe
PipedOutputStream	Output pipe
PrintStream	Output stream that contains **print( )** and **println( )**
PushbackInputStream	Input stream that supports one-byte "unget," which returns a byte to the input stream
RandomAccessFile	Supports random access file I/O
SequenceInputStream	Input stream that is a combination of two or more input streams that will be read sequentially, one after the other

Byte Stream class



Character Stream Class	Meaning
BufferedReader	Buffered input character stream
BufferedWriter	Buffered output character stream
CharArrayReader	Input stream that reads from a character array
CharArrayWriter	Output stream that writes to a character array
FileReader	Input stream that reads from a file
FileWriter	Output stream that writes to a file
FilterReader	Filtered reader
FilterWriter	Filtered writer
InputStreamReader	Input stream that translates bytes to characters
LineNumberReader	Input stream that counts lines
OutputStreamWriter	Output stream that translates characters to bytes
PipedReader	Input pipe
PipedWriter	Output pipe
PrintWriter	Output stream that contains **print( )** and **println( )**
PushbackReader	Input stream that allows characters to be returned to the input stream
Reader	Abstract class that describes character stream input
StringReader	Input stream that reads from a string
StringWriter	Output stream that writes to a string
Writer	Abstract class that describes character stream output

Console Input/ Output

- **Reading Console Input**:
- Console input is accomplished by reading from **System.in**.
- To obtain the character based stream attached to console, wrap System.in in a BufferedReader object. It supports buffered input stream.
- Most commonly used constructor.
  - BufferedReader(Reader inputReader)
- **Reader** is an abstract class. **InputStreamReader**, one of its concrete subclass converts byte to character.
- Constructor to obtain **InputStreamReader** object that is linked to **System.in**.
  - InputStreamReader(InputStream inputStream)
- System.in can be used for inputStream as it refers to object of type InputStream.

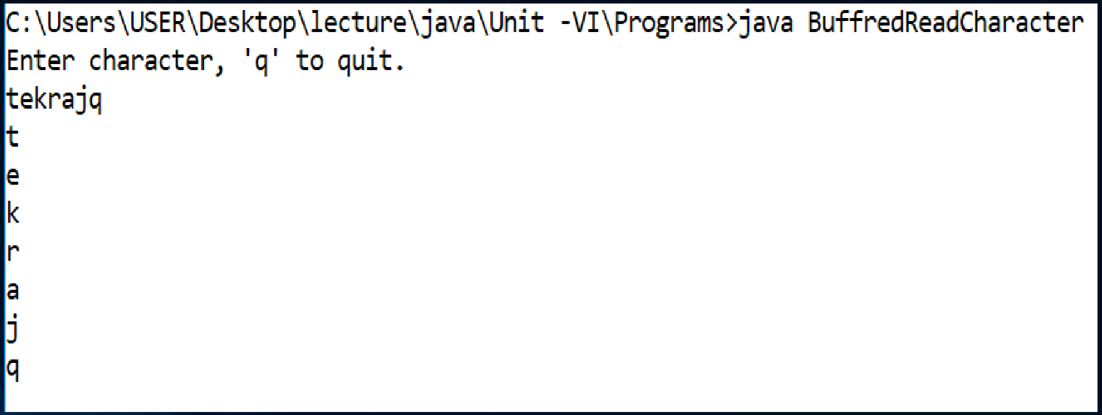
- Combining it all.

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

- br is a character based stream, linked to console through System.in.

- **Reading Characters:**
- To read character from BufferedReader, use read().
  - int read() throws IOException
- It reads the character from input stream and returns it as integer value.
- Returns -1 when end of stream is encountered.

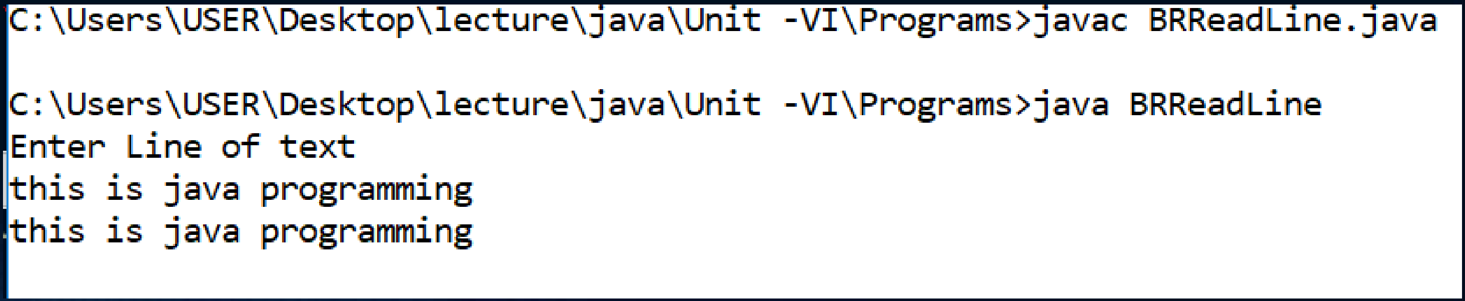
```
import java.io.*;
class BuffredReadCharacter{
        public static void main(String[] args) throws IOException {
                char c;
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("Enter character, 'q' to quit.");
                //read character and display
                do{
                        c = (char)br.read();
                        System.out.println(c);
                }while(c !='q');
        }
}
```



```
C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>java BuffredReadCharacter
Enter character, 'q' to quit.
tekrajq
t
e
k
r
a
j
q
```

- **Reading Strings**:
- To read character from BufferedReader, use readLine().
  - String readLine() throws IOException

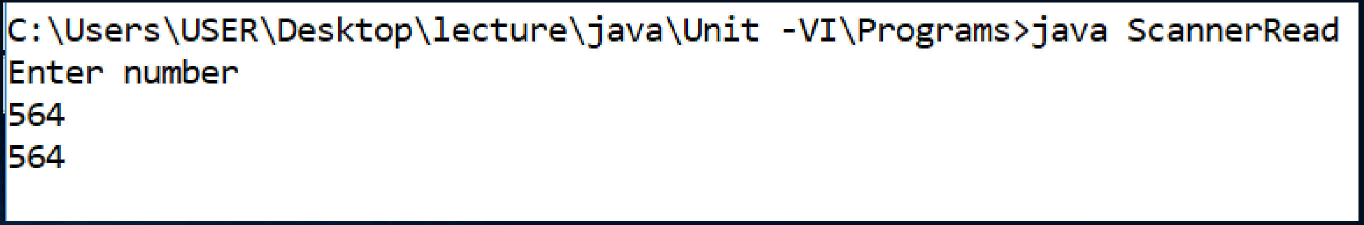
```java
import java.io.*;
class BRReadLine{
        public static void main(String[] args) throws IOException {
                String str;
                BufferedReader br = new BufferedReader(new 
InputStreamReader(System.in));
                System.out.println("Enter Line of text");
                str = br.readLine();
                System.out.println(str);

        }
}
```



```
C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>javac BRReadLine.java

C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>java BRReadLine
Enter Line of text
this is java programming
this is java programming
```

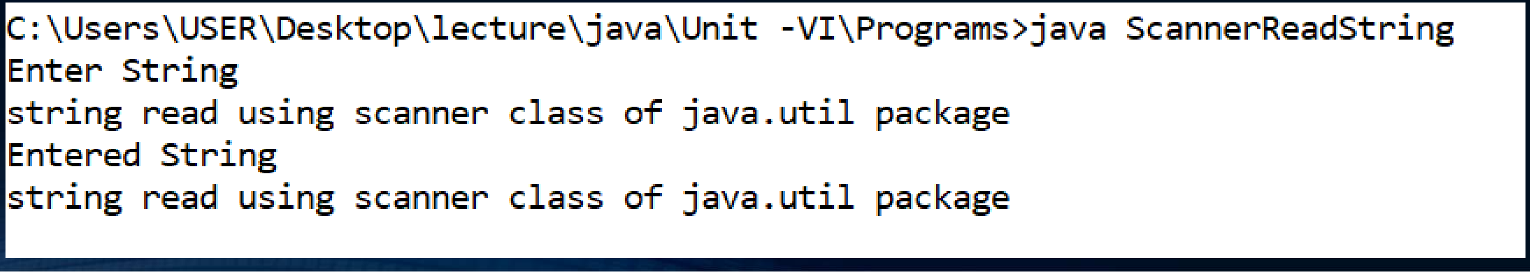
- **Reading using Scanner class:**
- The other way to read input from console is using the scanner class of java.util package.
- Scanner scanner = new Scanner(System.in);
- To read integer, use nextInt() as object.nextInt().
  - Eg: scanner.nextInt()
- To read string, use nextLine(); as object.nextLine().
  - Eg: scanner.nextLine();

```java
// reading integer using scanner class
import java.util.Scanner;
public class ScannerRead {
    public  static  void main(String [] args){
        int number;
        Scanner scanner = new Scanner( System.in );
        System.out.println("Enter number");
        number = scanner.nextInt();
        System.out.println(number);
    }
}
```



```
C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>java ScannerRead
Enter number
564
564
```

```java
// reading string using scanner class
import java.util.Scanner;
public class ScannerReadString {
    public static void main(String []args){
        String str;
        Scanner scanner = new Scanner( System.in );
        System.out.println("Enter String");
        str = scanner.nextLine();
        System.out.println("Entered String");
        System.out.println(str);
    }
}
```

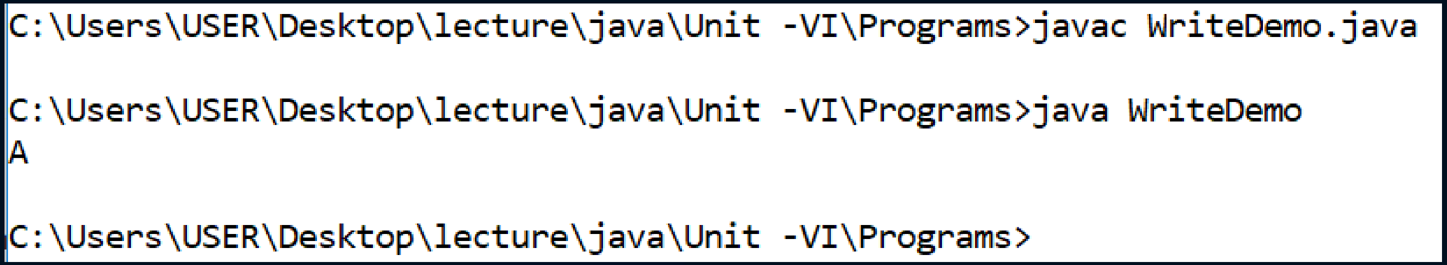


```
C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>java ScannerReadString
Enter String
string read using scanner class of java.util package
Entered String
string read using scanner class of java.util package
```

- **Writing Console Output:**
- Console output is most easily accomplished using print() and println().
- These methods are defined by class PrintStream, type of object referenced by System.out.
- **PrintStream** is an output stream derived from **OutputStream**, also implements low level method write().
- The method write() is also used to write to console.

```
void write(int byteval)
```

```java
public class WriteDemo {
    public static void main(String []args){
        int b;
        b = 'A';
        System.out.write(b);
        System.out.write('\n');
    }
}
```

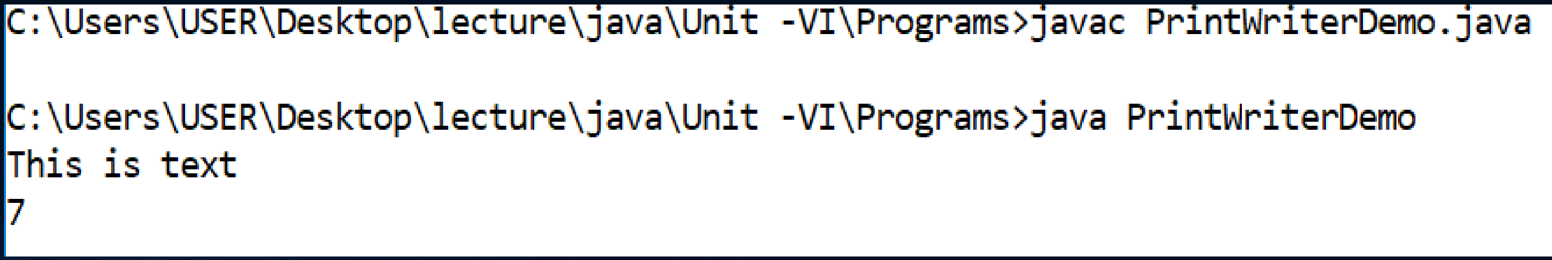


```
C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>javac WriteDemo.java

C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>java WriteDemo
A

C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>
```

- **PrintWriter Class**:
- **PrintWriter** stream is one of the character based class and recommended for real world application.
- One of the constructor of PrintWriter is

  PrintWriter(OutputStream outputStream, Boolean flushOnNewLine)
- outputStream is of type OutputStream .
- flushOnNewLine controls whether java flushes the output stream every time println() is called.
- If flushOnNewLine is true, flushing automatically takes place.
- If flushOnNewLine is false, flusing is not automatic.

```java
import java.io.*;
public class PrintWriterDemo {
   public static void main(String []args){
      PrintWriter pw = new PrintWriter(System.out, true);
      pw.println("This is text");
      int i = 7;
      pw.println(i);
   }
}
```



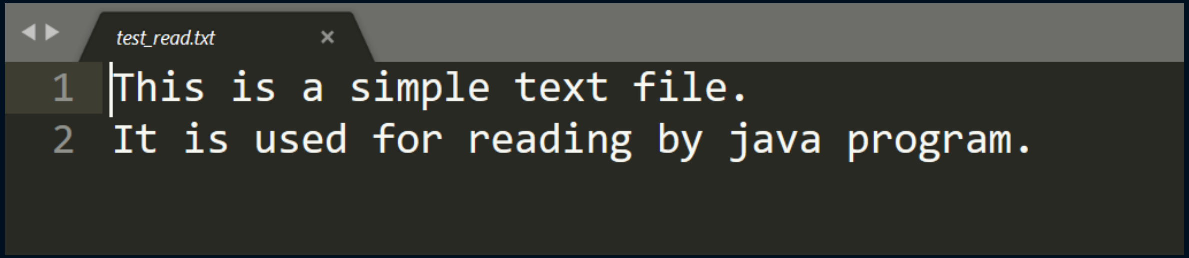
```
C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>javac PrintWriterDemo.java

C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs>java PrintWriterDemo
This is text
7
```

Reading and Writing files

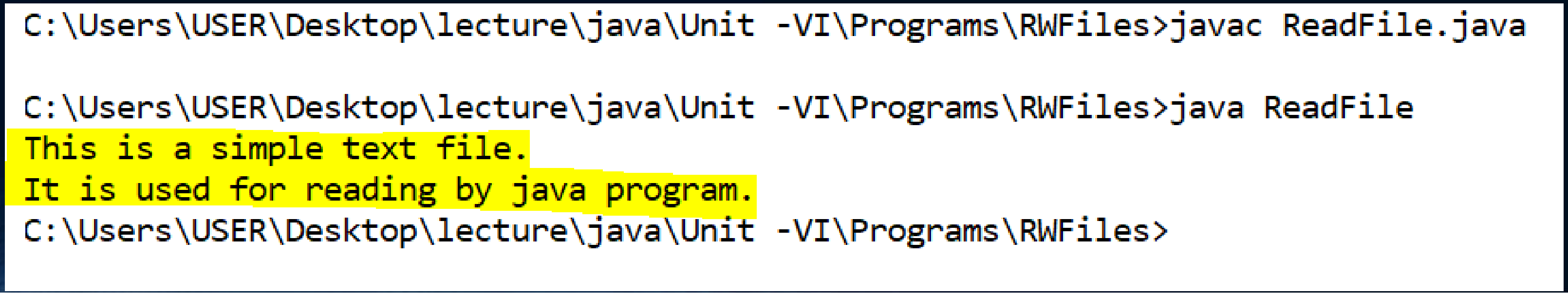
- Two stream classes **FileInputStream** and **FileOutputStream** are used to read and write new file.
- To open file, simply create an object of one of these classes specifying file name as argument to constructor as shown below.
  - FileInputStream(String filename) throws FileNotFoundException
  - FileOutputStream(String filename) throws FileNotFoundException
- When done with file, close it using close().
  - void close() throws IOException
- To read file we use read() defined within FileInputStream. Read returns -1 when end of file is reached.
  - int read() throws IOException

- To write file we use write() defined within FileOutputStream.
    - void write(int byteval) throws IOException
- Writes specified bytevalue to the file, declared as integer.

```java
import java.io.*;
class ReadFile{
    public static void main(String[] args) throws IOException{
        int i;
        FileInputStream fin;
        try{
            fin = new FileInputStream("test_read.txt");
            while((i = fin.read())!= -1){
                System.out.print((char)i);
            }
            fin.close();
        }catch(FileNotFoundException e){
            System.out.println("File not found");
            return;
        }
    }
}
```

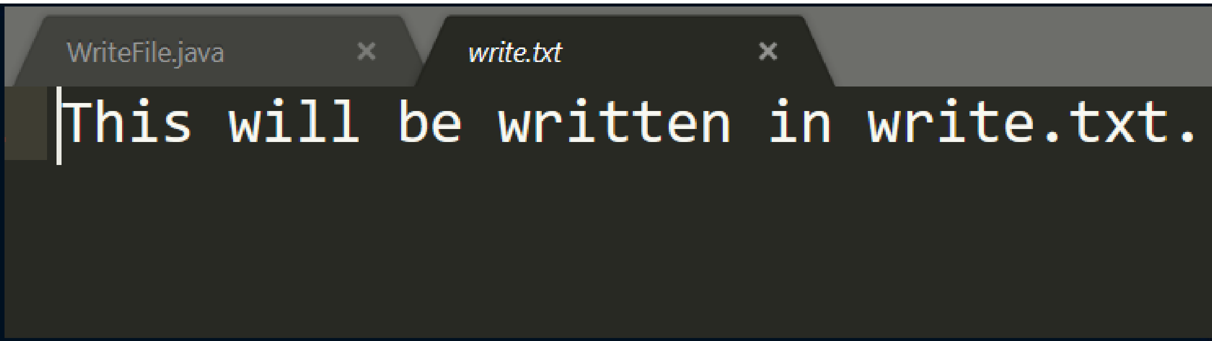


```
This is a simple text file.
It is used for reading by java program.
```

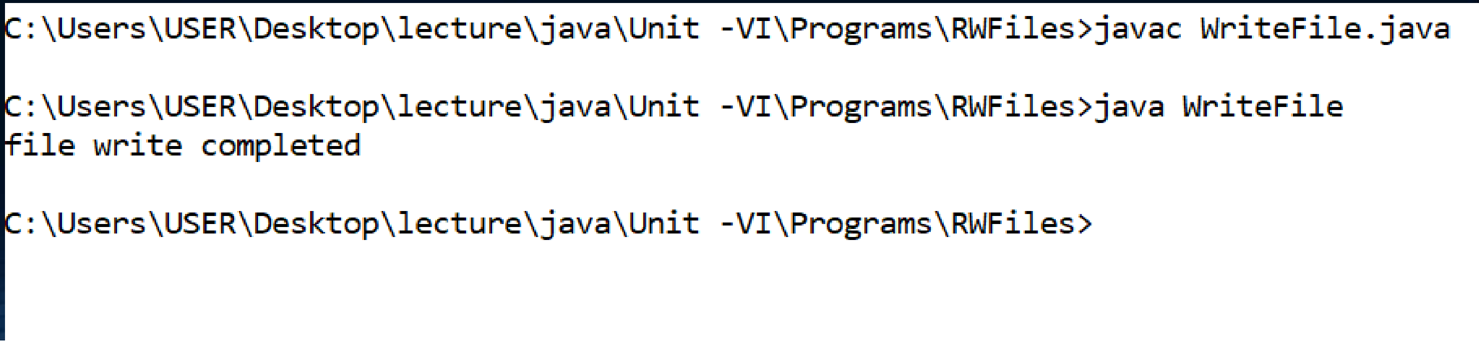


```
C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs\RWFiles>javac ReadFile.java

C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs\RWFiles>java ReadFile
This is a simple text file.
It is used for reading by java program.
C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs\RWFiles>
```

```java
import java.io.FileOutputStream;
public class WriteFile {
   public static void main(String args[]){
        try{
          FileOutputStream fout=new FileOutputStream("write.txt");
          String s="This will be written in write.txt.";
          byte b[]=s.getBytes();//converting string into byte array
          fout.write(b);
          fout.close();
          System.out.println("file write completed");
         }catch(Exception e){
                   System.out.println(e);
         }
     }
}
```



```
This will be written in write.txt.
```



```
C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs\RWFiles>javac WriteFile.java

C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs\RWFiles>java WriteFile
file write completed

C:\Users\USER\Desktop\lecture\java\Unit -VI\Programs\RWFiles>
```

Suggested Readings

- The respective topics in The complete Reference Java 7 (or any higher edition) by Hebert Schildt (P285-P296)
- Oracle official java documentation

References

- The complete Reference Java 7 by Hebert Schildt
- Java 8 in Action by Dreamtech press.
- Mit Opencourseware
- http://ee402.eeng.dcu.ie/
- https://www.javatpoint.com/
- https://docs.oracle.com/javase/tutorial/?sess=16e492aba13789410194of7f88d9f51f
- https://images.google.com for Images