OOP through JAVA (JNTUK-R19-2-1-ECE)
UNIT IV: I/O PROGRAMMING

Syllabus:

Text and Binary I/O, Binary I/O classes, Object I/O, Random Access Files. Event driven model, handling events

1. TEXT AND BINARY I/O:

- Text files are more readable by humans
- Binary files are more efficient
  - computers read and write binary files more easily than text
- Java binary files are portable
  - they can be used by Java on different machines
  - Reading and writing binary files is normally done by a program
  - text files are used only to communicate with humans

Java Text Files

- Source files
- Occasionally input files
- Occasionally output files

Java Binary Files

- Executable files (created by compiling source files)
- Usually input files
- Usually output files

Example:

- Number: 127 (decimal)
  - Text file
    - Three bytes: "1", "2", "7"
    - ASCII (decimal): 49, 50, 55
    - ASCII (octal): 61, 62, 67
    - ASCII (binary): 00110001, 00110010, 00110111
  - Binary file:
    - One byte (`byte`): 01111110
    - Two bytes (`short`): 00000000 01111110
    - Four bytes (`int`): 00000000 00000000 00000000 01111110

i) TEXT FILE I/O:

- Important classes for text file output (to the file)

PrintWriter
FileOutputStream [or **FileWriter**]

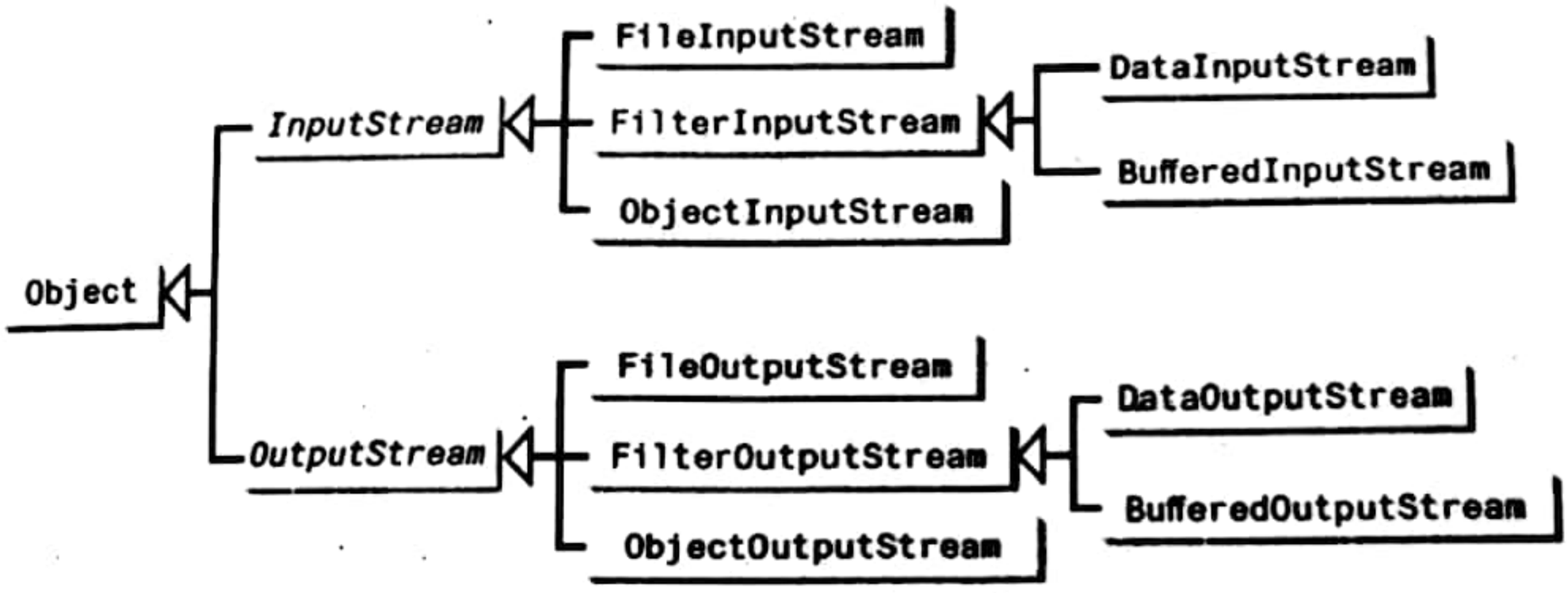
- Important classes for text file input (from the file):

BufferedReader
FileReader

- **FileOutputStream** and **FileReader** take file names as arguments.
- **PrintWriter** and **BufferedReader** provide useful methods for easier writing and reading.
- Usually need a combination of two classes
- To use these classes your program needs import java.io.*;

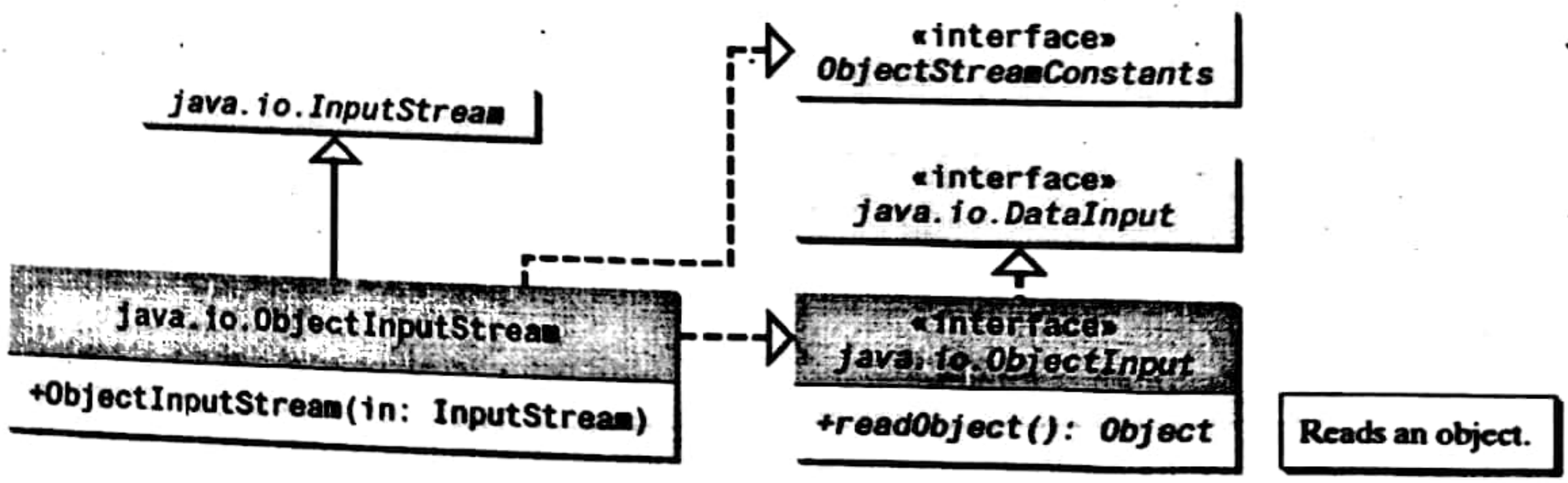
2. BINARY I/O CLASSES:

- The abstract InputStream is the root class for reading binary data, and the abstract OutputStream is the root class for writing binary data.
- InputStream, OutputStream, and their subclasses are for performing binary I/O is depicted as,

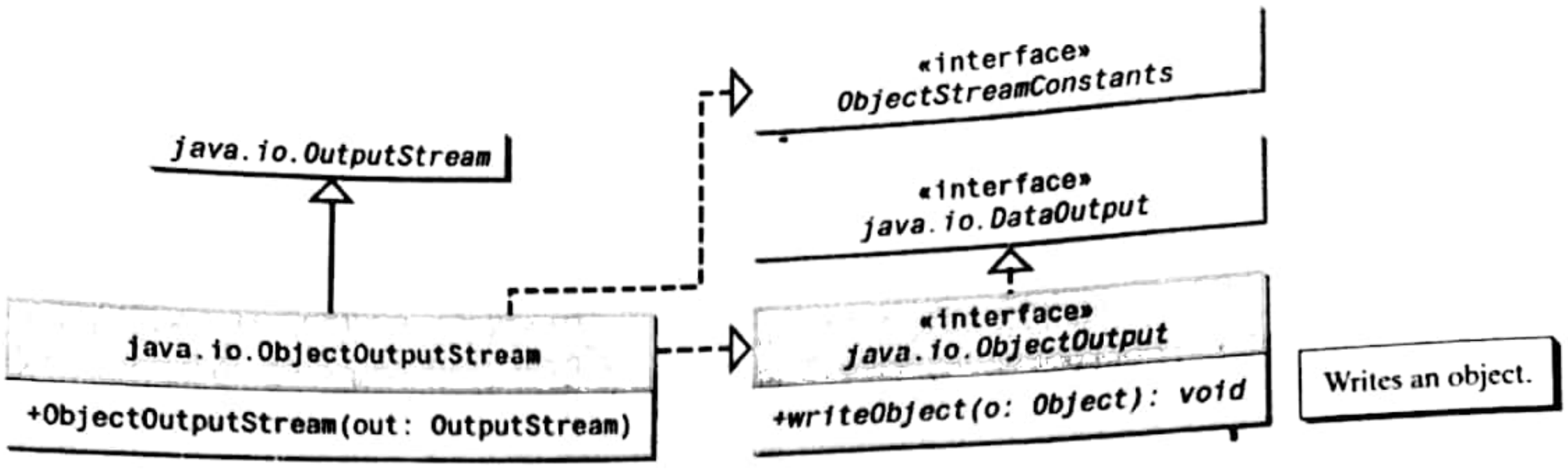


3. OBJECT I/O:

- ObjectInputStream/ ObjectOutputStream classes can be used to read/write serializable objects.
- DataInputStream/ DataOutputStream enables you to perform I/O for primitive-type values and strings.
- ObjectInputStream/ ObjectOutputStream enables you to perform I/O for objects in addition to primitive-type values and strings.
- Since ObjectInputStream/ ObjectOutputStream contains all the functions of DataInputStream/ DataOutputStream, you can replace DataInputStream/DataOutputStream completely with ObjectInputStream/ ObjectOutputStream.
- ObjectInputStream can read objects, primitive-type values, and strings is depicted as,



- ObjectOutputStream can write objects, primitive-type values, and strings is depicted as,



4. RANDOM ACCESS FILES:

- This class is used for reading and writing to random access file.
- A random access file behaves like a large array of bytes.
- There is a cursor implied to the array called file pointer, by moving the cursor we do the read write operations.
- If end-of-file is reached before the desired number of byte has been read than EOFException is thrown.
- It is a type of IOException.

Constructors:

Constructor	Description
RandomAccessFile(File file, String mode)	Creates a random access file stream to read from, and optionally to write to, the file specified by the File argument.
RandomAccessFile(String name, String mode)	Creates a random access file stream to read from, and optionally to write to, a file with the specified name.

Methods:

Modifier and Type	Method	Method
void	close()	It closes this random access file stream and releases any system resources associated with the stream.
FileChannel	getChannel()	It returns the unique FileChannel object associated with this file.
Int	readInt()	It reads a signed 32-bit integer from this file.
String	readUTF()	It reads in a string from this file.

Void	seek(long pos)	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.
Void	writeDouble(double v)	It converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the file as an eight-byte quantity, high byte first.
Void	writeFloat(float v)	It converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the file as a four-byte quantity, high byte first.
Void	write(int b)	It writes the specified byte to this file.
Int	read()	It reads a byte of data from this file.
Long	length()	It returns the length of this file.
Void	seek(long pos)	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.

Example:

```
import java.io.IOException;
import java.io.RandomAccessFile;

public class RandomAccessFileExample {
   static final String FILEPATH ="myFile.TXT";
   public static void main(String[] args) {
      try {
         System.out.println(new String(readFromFile(FILEPATH, 0, 18)));
         writeToFile(FILEPATH, "I love my country and my people", 31);
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
private static byte[] readFromFile(String filePath, int position, int size)
      throws IOException {
   RandomAccessFile file = new RandomAccessFile(filePath, "r");
   file.seek(position);
   byte[] bytes = new byte[size];
   file.read(bytes);
   file.close();
```

```
        return bytes;
    }
    private static void writeToFile(String filePath, String data, int position)
            throws IOException {
        RandomAccessFile file = new RandomAccessFile(filePath, "rw");
        file.seek(position);
        file.write(data.getBytes());
        file.close();
    }
}
```

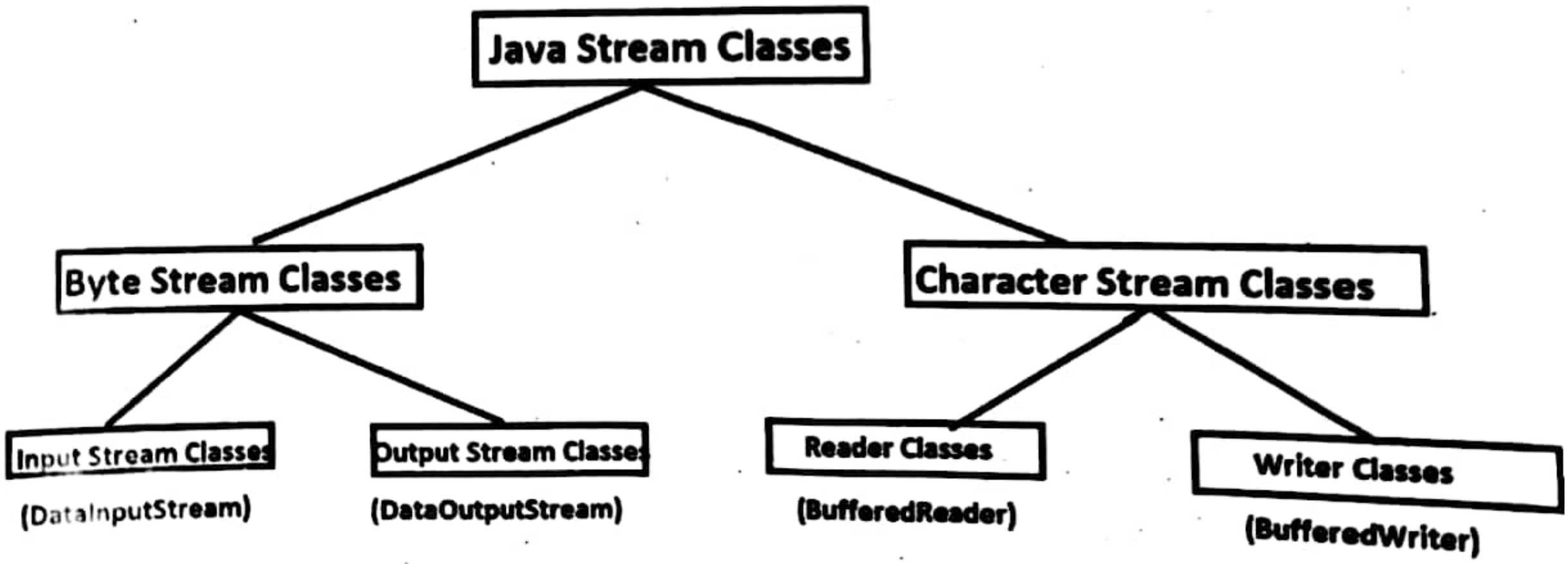
- The myFile.TXT contains text "This class is used for reading and writing to random access file.
- after running the program it will contains,
  This class is used for reading I love my country and my peoplele.

5. EVENT DRIVEN MODEL - HANDLING EVENTS:

REFER UNIT-3 TOPIC-4 (PAGE-63)

6. Managing I/O files in Java:

- A Stream in java is a path along which data flows.
- It has a source and destination
- Both the source and destination may be physical devices or programs or other streams in the s program.



7. Giving Input in Java:

- Java has various ways to read input from the keyboard. Some of them are,
  i) Direct Method
  ii) Using Command line Arguments
  iii) Using DataInputStream class
  iv) Using BufferedReader class

v) Using Scanner class

i) Direct Method:

- In this method, we will not use any input method.
- Directly we are initializing the input values at the time of declaration.

Example Program:

```
class Sum
 {
 public static void main(String args[])
  {
  int a=10,b=20;
   System.out.println("Sum of two numbers="+ (a+b));
  }
 }
```

Output:

Sum of two numbers=30

ii) Using Command line Arguments:

- The arguments which are passed in the execution line while executing a program.
- To receive the command line values, the main() contains a parameter of type string array.
- Then it reads the command line values in the form of strings.
- So we convert those strings into primitive type by using the concept called wrapping.

Example Program:

```
class Sum2
 {
 public static void main(String args[])
  {
      int a,b;
      a=Integer.parseInt(args[0]);
      b=Integer.parseInt(args[1]);
      System.out.println("Sum of two numbers="+ (a+b));
  }
 }
```

Execution:

```
>javac  sum.java
>java  sum  10  20
```

Output:

Sum of two numbers=30

iii) Using DataInputStream class

- Java can take inputs in the form of strings only.
- The keyboard gives input in the form of bytes.
- The DataInputStream class object reads input in bytes.

Syntax:

```
DataInputStream dis=new DataInputStream(System.in);
```

- A method called readLine() takes the bytes from DataInputStream class object and stores in the form of strings represented like,

dis.readLine()

- So we convert those strings into primitive type by using the concept called wrapping.

Example Program:

```
import java.io.*;
class Sum
 {
  public static void main(String args[]) throws IOException
   {
      int a,b;
      DataInputStream dis=new DataInputStream(System.in);
      System.out.println("Enter values for a,b");
      a=Integer.parseInt(dis.readLine());
      b=Integer.parseInt(dis.readLine());
      System.out.println("Sum of two numbers="+ (a+b));
   }
 }
```

Output:

```
Enter values for a,b
10
20
Sum of two numbers=30
```

iv) Using BufferedReader class:

- Java can take inputs in the form of strings only.
- The keyboard gives input in the form of bytes.
- The BufferedReader class object reads input only in characters.
- So we require a mediator class (InputStreamReader class) to convert bytes into charcters.
- The following steps are required to give input using BufferedReader class,

Step1: Attach Keyboard to Mediator class

Step2: Attach Mediator class to BufferedReader class

Syntax:

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

- BufferedReader class has a method called readLine(), it is responsible to read any value from th keyboard and stores in the form of strings.
- readLine() will fetch an entire string upto space given or new line.

Example Program:

```
import java.io.*;
class Sum
 {
  public static void main(String args[]) throws IOException
   {
      int a,b;
      BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
```

```
    System.out.println("Enter values for a,b");
    a=Integer.parseInt(br.readLine());
    b=Integer.parseInt(br.readLine());
    System.out.println("Sum of two numbers="+ (a+b));
  }

}
```

Output:

```
Enter values for a,b
10
20
Sum of two numbers=30
```

v) Using Scanner class:

- Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double etc. and strings.
- It is the easiest way to read input in a Java program.
- It is from the java.util package.
- The Java Scanner class breaks the input into tokens using a delimiter that is whitespace by default.
- It provides many methods to read and parse various primitive values.
- Java Scanner class is widely used to parse text for string and primitive types using a regular expression.

Example Program:

```
import java.util.Scanner;
class Sum
{
  public static void main(String args[])
  {
    int a,b;
    System.out.println("Enter values for a,b");
    Scanner in=new Scanner(System.in);
    a=in.nextInt();
    b=in.nextInt();
    System.out.println("Sum of two numbers=" + (a+b));
  }
}
```

Output:

```
Enter values for a,b
10
20
Sum of two numbers=30
```

8. Wrapper classes in Java:

- **Wrapper class in java provides the mechanism *to convert primitive types into objects and objects into primitive types.***

- Converting a primitive value which is in the form of a string into its primitive type is called Wrapping.
- To do Wrapping, each primitive type is having a class called Wrapper class.
- Wrapper classes contains a common static method called parseXXX() method and it is used for casting or conversion.
- These Wrapper classes are available in java.lang package.
- The different wrapper classes for the primitive data types given in a table as,

Primitive Type	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
Int	Integer
Long	Long
Float	Float
double	Double