Unit 5 : Using I/O

5.1 Console and file I/O

Input/output (I/O) is used to receive data from the user and display or store results.

Console I/O

Console I/O means reading input from the keyboard and displaying output on the screen.

↳ System.out.println() method is used to print output with a new line

↳ System.out.print() ⇒ print output without a new line.

↳ System.in → used to take input from keyboard.

↳ Scanner class is used to accept data entered by the user which is belongs to the java.util package.

Example

```
import java.util.Scanner;
public class inputexample
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);
    System.out.print("enter your name: ");
    String name = sc.nextLine();
    System.out.println("Hello" + name);
```

3?

output:

Enter your name: Rajesh

Hello Rajesh

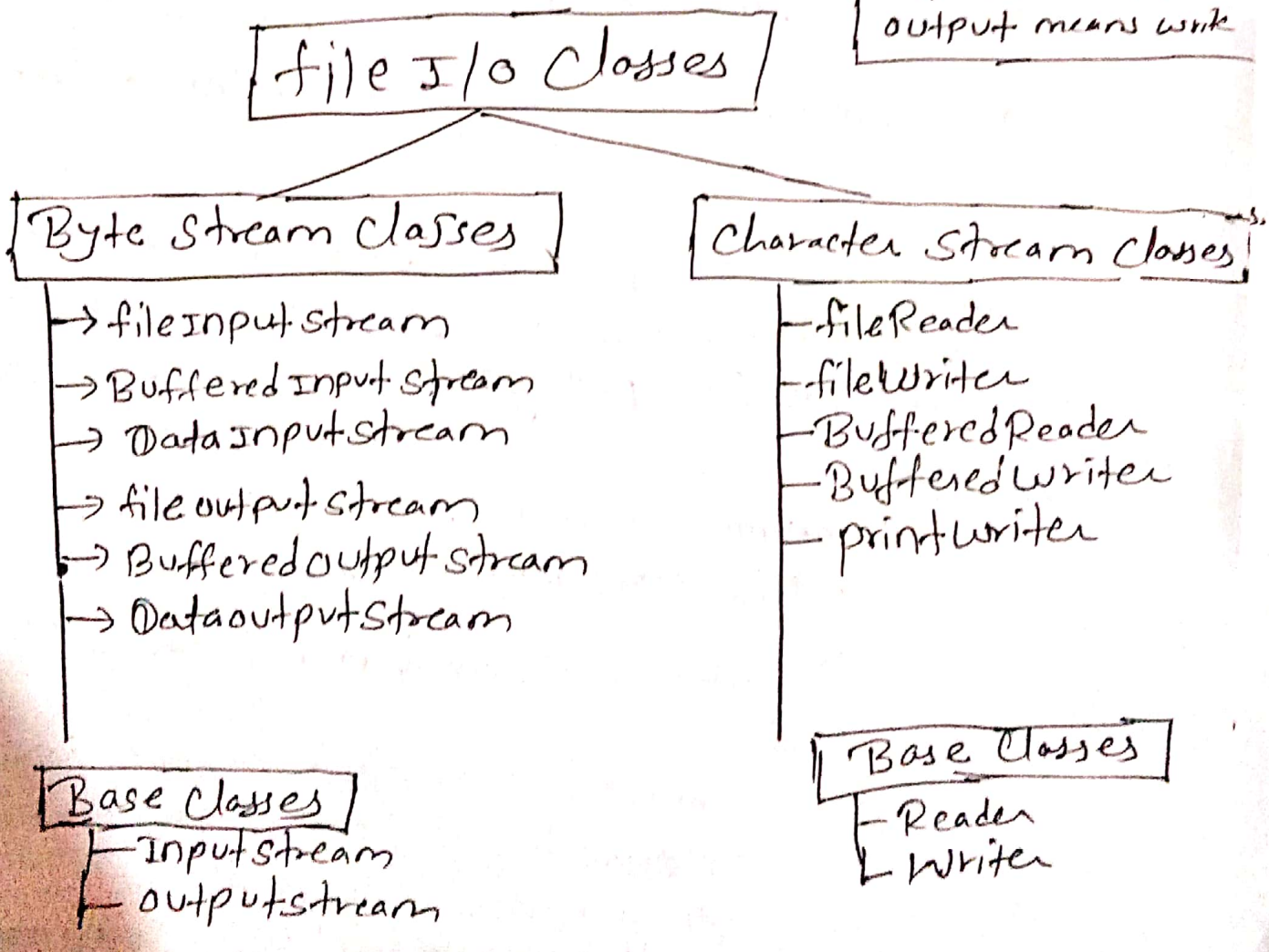
file I/O

file I/O is used to store data in files and read data from files.

Java provides file input/output (I/O) classes in the java.IO package to read and write data from files.

These classes are mainly classified into two types

Note:
input means Read
output means write



5.2 opening and closing files

Before reading or writing data, a file must be opened.

opening a file: files can be opened using classes like:

- → fileInputStream
- → fileOutputStream
- → fileReader
- → fileWriter

Example of opening a file in java

```
fileInputStream file = new fileInputStream("data.txt");
```

closing a file:- After finishing operations, the file should be closed to free system resources.

example

```
file.close();
```

closing files prevents data loss and memory leaks.

Q. Program example in java to open and close the file

```
import java.io.*;
public class fileopenclose
{ public static void main(String[] args) throws Exception
  {
    fileInputStream file = new fileInputstream("data.txt");
    file.close();
  }
}
```

Q. open and close file using try-catch

```
import java.io.*;
public class fileExample
{
  public static void main(String[] args)
  {
    try
    {
      FileReader fr = new FileReader("data.txt");
      System.out.println("file opened successfully");
      fr.close();
      System.out.println("file closed successfully");
    }
    catch (Exception e)
    {
      System.out.println("Error: " + e);
    }
  }
}
```

5.3 Scanner class

The Scanner class is used to take input from the user. It belongs to java.util.scanner package.

Syntax to Creating Scanner object

```
Scanner sc = new Scanner(System.in);
int num = sc.nextInt();
```

methods for different datatypes

nextInt() → Reads integer
nextDouble() → Reads double
next() → Read single word
nextLine() → Reads full line.

5.4 Byte Streams and Character Streams

Java I/O streams are divided into two types:-

S.No	Byte Streams	Character Streams
①	Handles binary data (8-bit bytes)	Handles text data (16-bit characters)
②	Used for images, audio, video and binary files.	Used for text files (.txt)
③	Classes inherit from InputStream / OutputStream	Classes inherit from Reader / Writer
④	Reads / write data byte by byte.	Reads / writes data character by character.
⑤	Example Classes: FileInputStream, FileOutputStream, BufferedInputStream BufferedOutputStream	Example Classes: FileReader, FileWriter, BufferedReader, BufferedWriter
⑥	cannot directly handle unicode text	Can handle unicode text properly.
⑦	faster for binary data.	optimized for text data.

5.5 Reading and writing Byte Streams

Byte Streams read and write binary data.