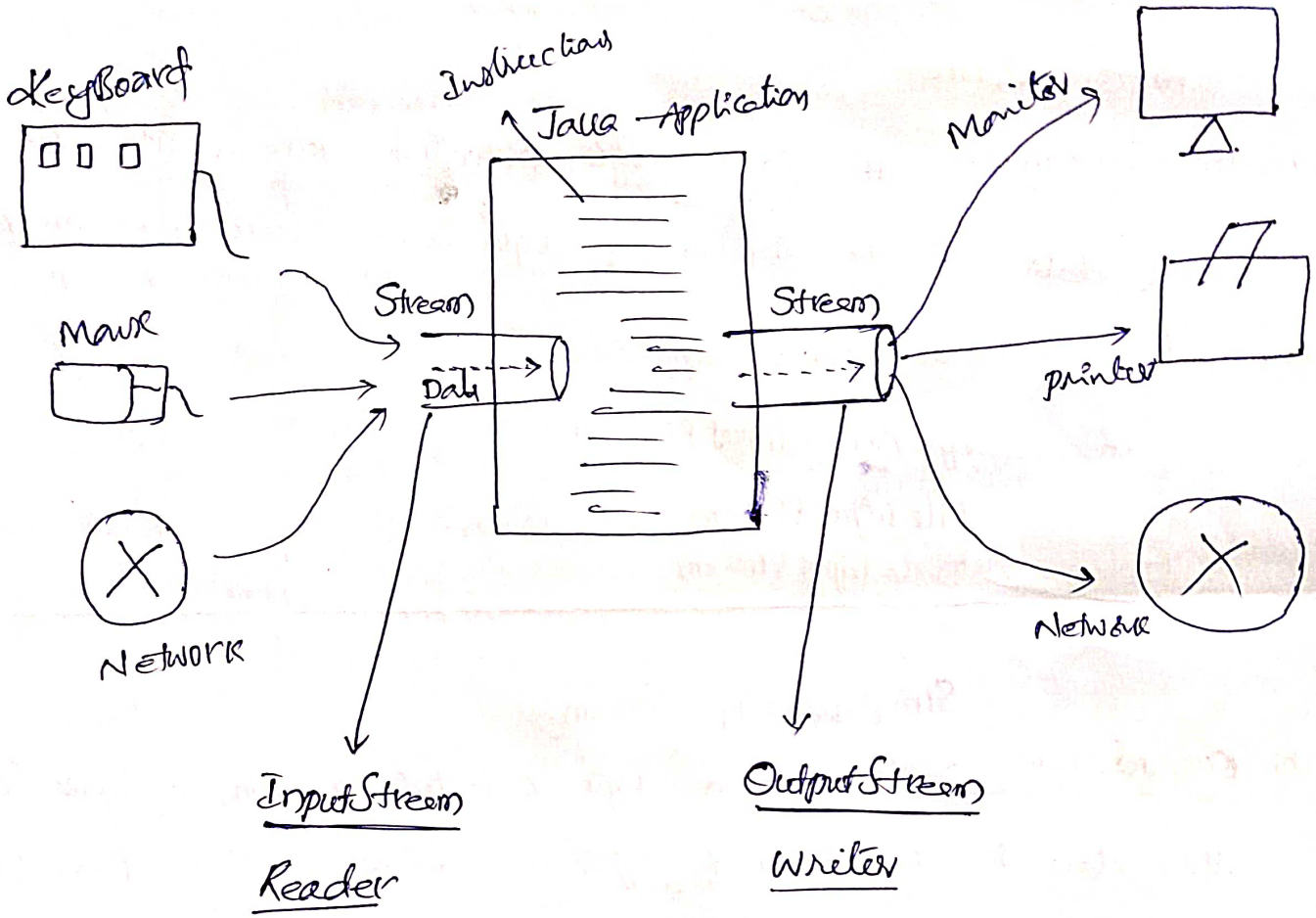
IO Streams :-

Stream : is a channel (or) medium, it able to allow data in continuous flow from input devices to java applications and from java applications to output Devices.



Types of streams :-

1. Byte Oriented streams
2. character oriented streams

It able to allow data in [illegible]
from input devices to java application and from java application
to output devices

a) InputStream

b) OutputStream

(a) InputStream :- It is a byte oriented stream, it able to allow data in the form of bytes in continuous flow from input devices to Java application

Ex:- ByteArrayInputStream ✓
FileInputStream ✓
DataInputStream ✓
ObjectInputStream ✓
StringBufferInputStream ✓

(b) OutputStream :- It is a byte oriented stream, it able to allow data in the form of bytes in continuous flow from java applications to output devices

Ex:- ByteArrayOutputStream
FileOutputStream
DataOutputStream
ObjectOutputStream
StringBufferOutputStream
PrintStream

Note :- In byte oriented streams each and every data length is 1 byte.

Note :- In Byte oriented streams, all stream classes are ended with "Stream".

2. Character Oriented Stream :-

It able to allow data in continuous flow in the form of characters from input devices to Java application and from java application to output devices.

(a) Reader.

(b) Writer.

(a) Reader :- It able to carry data in the form of characters in continuous flow from input devices to java applications.

Ex:- CharArrayReader.
FileReader
BufferedReader
InputStreamReader
StringBufferReader
- - - - -

(b) Writer :- It able to allow data in the form of characters in continuous flow from java applications to output devices.

Ex:- CharArrayWriter
FileWriter
BufferWriter
PrintWriter.
- - - - -
- - - - -

Note :- In character oriented streams, each and every data item length is '2' bytes.

Note :- In character oriented streams all stream classes are ended either Reader (or) with Writer

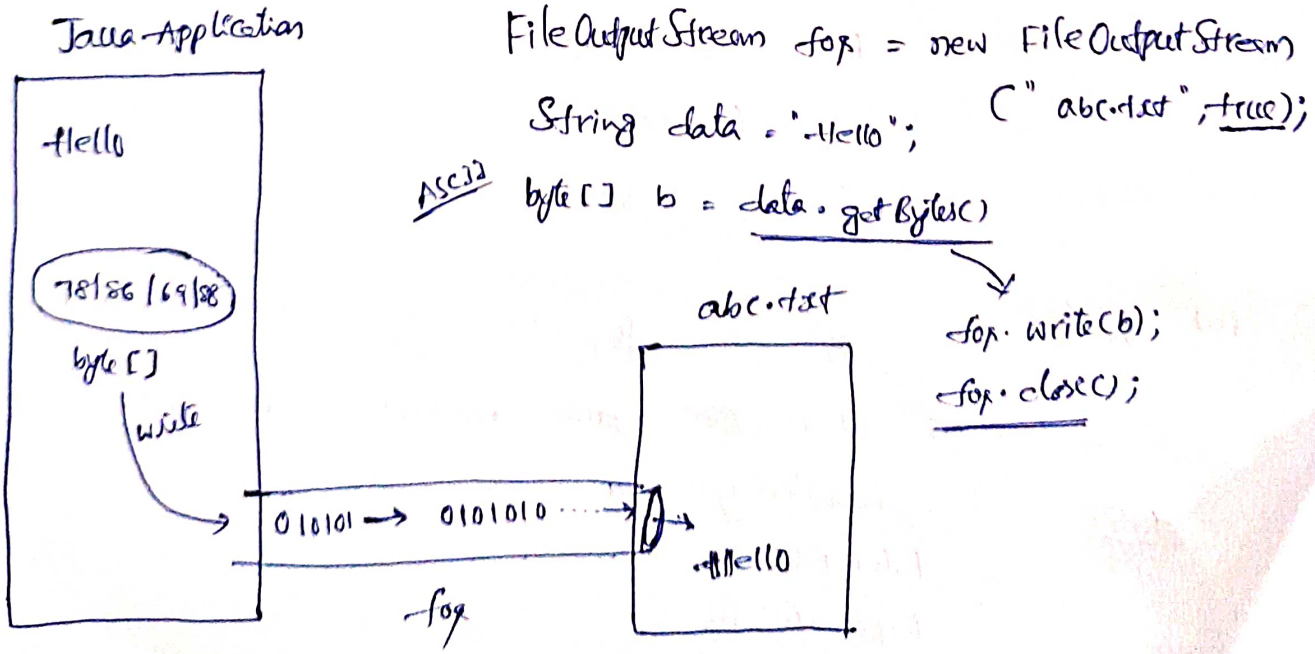
=> JAVA has declared all the streams in the form of predefined classes in package "java.io" package.
↳ unlike user defined exceptions and Threads

File operations :-

1. File Output Stream
2. File Input Stream
3. File Writer
4. File Reader.

1. FileOutputStream :-

It able to carry data from java application to particular target file.



FileOutputStream fos = new FileOutputStream ("abc.txt", true);
String data = "Hello";
ASCII byte[] b = data.getBytes()
fos.write(b);
fos.close();

```
import java.io.*;
class FileDemo
{
    public static void main(String[] args) throws Exception
    {
        FileOutputStream fos = new FileOutputStream("doc.txt", true);
        String data = "Durga Software Solutions";
        byte[] b = data.getBytes();
        fos.write(b);
        fos.close();
    }
}
```

append new data to old data.

By default false

mention path if target available at another location.

E:/abc/xyz/abc.txt → forward → file separator.

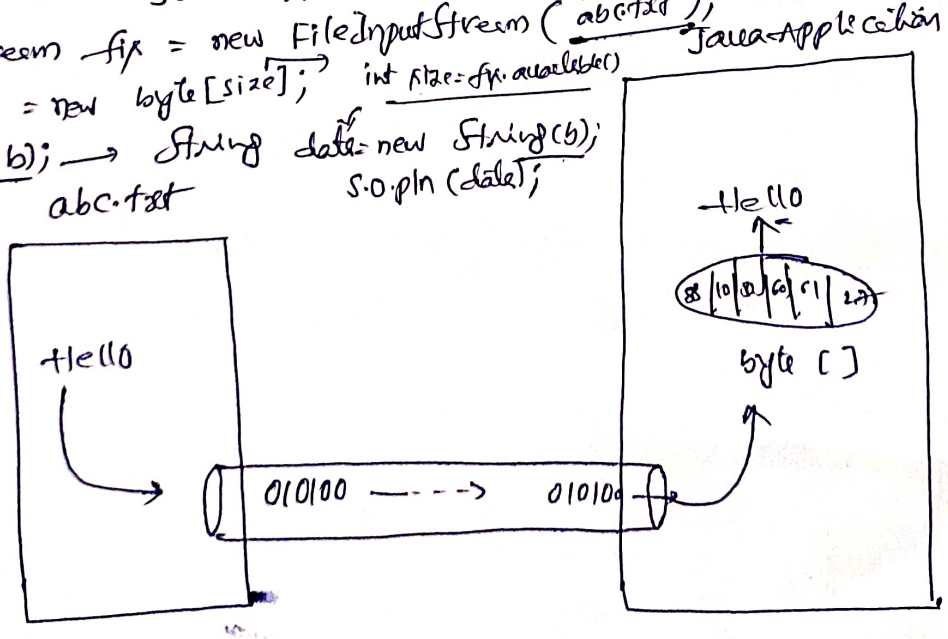
windows command prompt → supports Backward

2) FileInputStream :-

It able to carry data from a particular source → JVM will search for file

file to java application.

```
FileInputStream fis = new FileInputStream("abc.txt");
byte[] b = new byte[size];
fis.read(b);
int size = fis.available()
String data = new String(b);
S.O.println(data);
```



=> abc.txt file is not existed, JVM raises exception

java.io.FileNotFoundException →

=> when fis is created when specified file existed, immediately what ever data available data will come to fileinputstream (Internally Text to Binary)

```
import java.io.*;
class FileDemo
{
    public static void main(String args[]) throws Exception
    {
        String filename = args[0];
        FileInputStream fis = new FileInputStream(filename);
        int size = fis.available();
        fis.read(b); → byte[] b = new byte[size];
        String data = new String(b);
        System.out.println(data);
    }
}
```

write a java program to read and display data from a particular file by taking file name as command line input.

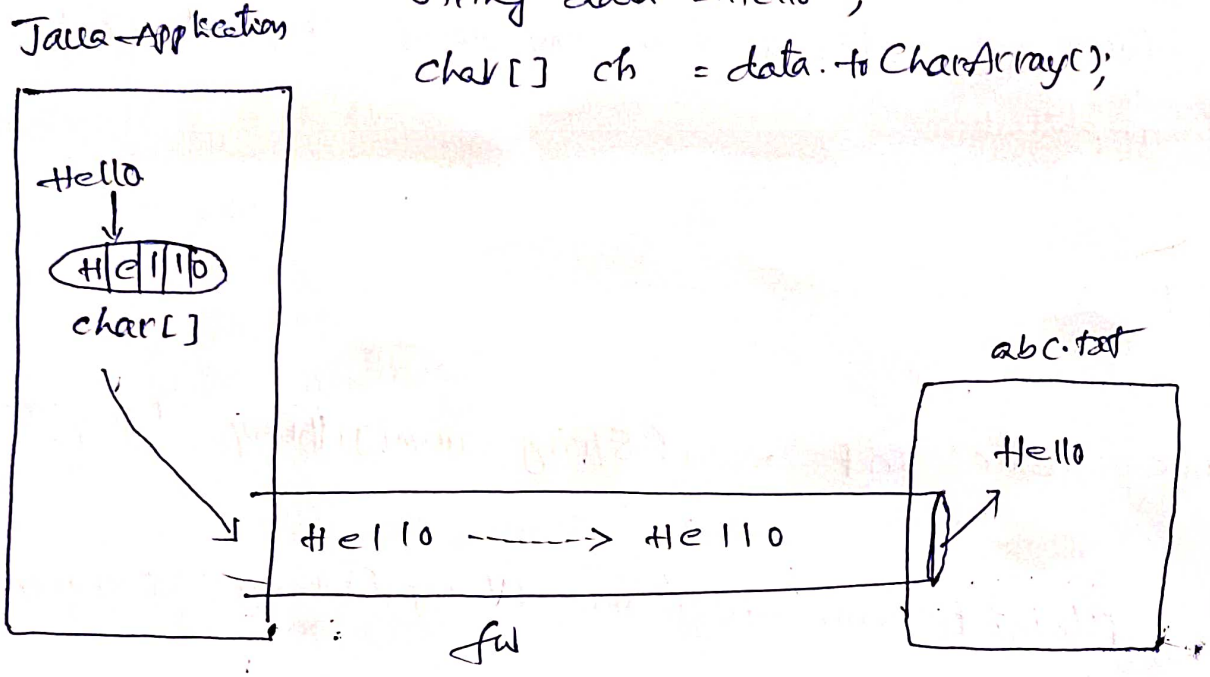
a) write a java program to read data from a particular file and calculate the no. of words which are existed in the source file and calculate how many no. of times the word "cse" is repeated.

```
class Demo
{
    public static void main(String args[]) throws Exception
    {
        FileInputStream fis = new FileInputStream("E:/abc/abc.txt");
        int size = fis.available();
        byte[] b = new byte[size];
        fis.read(b);
        String data = new String(b);
        System.out.println();
        String[] str = data.split(" ");
        int size = str.length;
        S.o.pln("No. of words : " + size);
        int count = 0;
        for(int i=0; i<size; i++)
        {
            String val = str[i];
            if(val.equals("cse"));
            {
                count = count + 1;
            }
        }
        S.o.pln("cse is repeated " + count);
```

FileWriter :-

It able to carry data from java application to a particular target file.

```
FileWriter fw = new FileWriter("abc.txt", true);
String data = "Hello";
char[] ch = data.toCharArray();
```

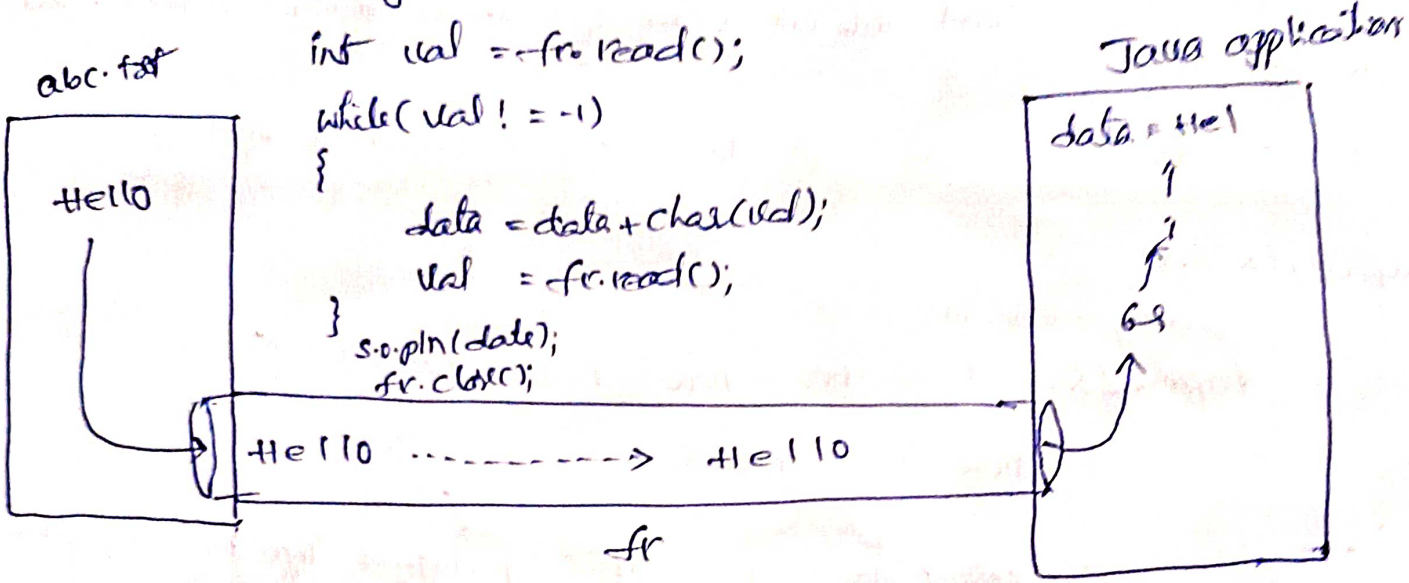



```
import java.io.*;
class FileWriterDemo
{
    public static void main(String[] args) throws Exception
    {
        FileWriter fw = new FileWriter("abc.txt");
        String data = "VVIT-VIVA";
        char[] ch = data.toCharArray();
        fw.write(ch);
        fw.close();
    }
}
```

FileReader :-

It able to carry data from a particular path file to java application.

```
FileReader fr = new FileReader("abc.txt");
String data = "";
int val = fr.read();
while(val != -1)
{
    data = data + char(val);
    val = fr.read();
}
s.o.pln(data);
fr.close();
```




```
class FileReaderDemo
{
    public static void main(String args[]) throws Exception
    {
        FileReader fr = new FileReader("abc.txt");
        String data = "";
        int val = fr.read();
        while(val != -1)
        {
            data = data + (char) val
            val = fr.read();
        }
        System.out.println(data);
    } fr.close();
}
```

Dynamic Input Approaches :-

providing input to the java application at runtime is called as Dynamic input.

1. BufferedReader → Java BufferedReader class is used to read the text from a character based input stream. It can be used to read data line by line by readLine(). It enhances performance fast.
2. Scanner
3. Console → To improve performance of input operations.

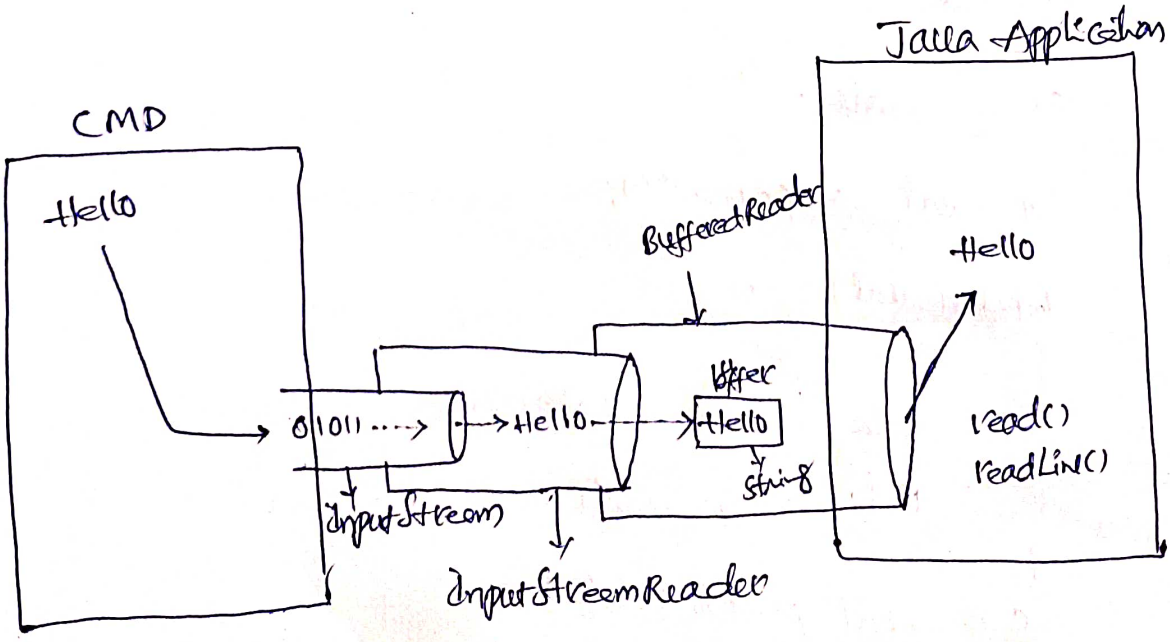
1. BufferedReader :-

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

- BufferedReader → To convert data from Bytes representation to char representation.
- InputStreamReader → refer predefined InputStream obj which is connected with CMD

⇒ read() method is able to read a single character in the form of its ASCII value. public int read()

⇒ readLine() method is able to read line of data in the form of String directly. public String readLine()



readLine()

1) Read data in the form of String by using readLine()

2) convert that data from String type to respective primitive type by using wrapper classes.

Scanner

To get primitive data directly with out using wrapper classes.

java.util. package in JDK 5.0 version

```
Scanner s = new Scanner(System.in);
public String next();
public String nextLine();
public xxx nextXxx(): ⟶ primitive data directly.
```

*** What is the Requirement to use Console over BufferedReader and Scanner?

Ans: Drawbacks of BufferedReader and Scanner

⇒ for every dynamic input we have to write two instructions

1. S.O.pln(----) to display request message.
2. readline() or next() / nextXxx() to read dynamic input

→ No security for dynamic input data like password data, PIN Numbers, Secrete codes..... when we enter these types of data on command prompt as dynamic input then that data is visible in screen, we are unable to hide that in BR and Scanner.

→ To overcome these problems we have to use

"java.io.Console"

```
import java.io.*;
class ConsoleDemo
{
    public static void main(String args[])
    {
        Console c = System.console();
        String uname = c.readLine("user name");
        char[]
        String pwd = c.readPassword("password:");
        String spwd = new String(pwd);
        if(uname.equals("vvit") && spwd.equals("vvit"))
        {
            S.o.println("user login is success");
        }
        else
        {
            S.o.println("user Login is failure");
        }
    }
```