#

```
interface i1{
        void m1();
    }

interface i2{
        void m2();
    }

class imple implements i1, i2
    {
public void m1(){                    //CTE

        sop("M1Method");
    }
}

class interface Test
{
  psv main(String[] args)
        {
          Imp i = new imp();
          i.m1();
          i.m2();
        }
}
```

Java.lang.String

```
Class Outer{
    int x=10;
    Class Inner
    {
        int y= 20;
        void m1()
        {
            SOP("MI inner"+x);
        }
    }
    void m2
    {
        int y;
    }
}
Class InnerTest
{
    PSV main(String[] args)
    {
        Outer O = new Outer();
        O.m2();
    }
}
```

Anonymous class:

```
interface i1
{
    void m1();
}

class Anonymous1
{
    PSV main(St
    {
    i1 i = new i1()
        {
        public void m1()
            {
            SOP('');
            }
        };
    }
}
```

Interface में कितने class create कर सकते हैं
2 से अधिक Ans= Yes

IMP

Exception Handling :-
- Require for graceful execution
- To generate user friendly o/p

Unexpected ~~problem~~ error occur in java

IMP (RTE) (Runtime error)

JVM will give the message

↳ Abnormal termination of prog
↳ Technical message which is not understandable by end user

⇒ Exception is an object which contains the Message ~~other~~ by System generated ~~error~~

In Java we have keyword to handle exceptions :-

→ try
→ catch
→ finally
→ throw
→ throws

try — write the suspect block

~~catch — has~~

catch — handle the exception

API (Application Programming Interface) — the package (Predefined Classes)

⇒ java.lang.Throwable is the superclass of all class in exception handling

java.lang.Exception (subclass)

- IE (Interrupted Ex)
- CNFE (Class not found)
- FileNotFoundEx
- RTE (Unchecked Ex)
  - NullPointerEx
  - AIOOBE (ArrayIndexOutOfBound)
  - NASE (~~Negative Array Size Ex~~)
  - AE (arithmetic Ex)
  - NFE (Number Format Ex)

(Checked Exception)

try :-

```
try {
        Line 1
        Line 2 ——— Exception occur
        Line 3
        Line 4
} catch (Arithmetic Ex. e) {

}
```

eg

```
class ExTest {
        void m1()
        {   sop("step1");
            sop("step2");
            sop("step3");   int i=10/0; // RTE
            sop("step4");
            sop("step5");
        }
}
class ExceptionTest {
    PSV main(String[] args)
    {
        ExTest t1 = new ExTest();
        t1.m1();
    }
}
```

A

A, K, M, N, R (S) V, W, X, Y, Z

```
main()
  {
    m1();
  }

m1()
  {
    m2();
  }

m2()
  {
    m3();
  }

m3()
  {
    exception;
  }
```

Java will check M3 that exception is handle or not

अगर नहीं M2 में

उसमें भी नहीं तो M1

उसमें भी नहीं तो main

उसमें भी नहीं तो JVM will provide default exception handling

class {

method

try

{

}

} catch(ArithmeticException e)

{

}

}

#

We can use Multiple catch block

```
class ExceptionTest
{
  PSV main(String[] args)
  {
    int r=0;
    String s1 = args[0];
    String s2 = args[1];
    int no1 = Integer.parseInt(s1);
    int no2 = Integer.parseInt(s2);
    try
    {
      r = no1/no2;
    } catch(ArrayIndexOutOfBoundException e1)
    {
      SOP("_____");
    } catch(<- ->)
    {
      SOP("");
    }
}
```

If any exception occur that we donot know then we make

```
catch(Exception e)
{
  SOP("Bye");
}
```

Scanned by CamScanner

→ Flow of try & catch block :-

try {

—
—
—

E1 exception occur

} catch (E1 e1) {

}

catch (E2 e2) {

}

अगर same exception दो बार लिखेंगे तो error आएगा।

Exception को अगर ऊपर लिखेंगे तो Error

"Exception is a superclass that can't be written above subclass

try

{

try & catch के बीच में कोई intermediate code allowed नहीं है।

}

SOP("hi");

catch{

}

Finally

↳ special block

try block execute होना हो या

finally block execute होगा ही हो

eg =

```
try
{
}
block catch ( )
{
}
finally {
}
```

चाहे try & catch block execute हो या ना हो या

finally block will definately execute

Real time use of finallyblock :-

is the place where we can close the connection (Resource releasing code is written in finally block)

If there is presence of finally block after try block but not catch block, it will execute.

~~#~~ We can also write try & catch block in finally block.

# try {} finally {}	try {} finally {} catch {}	try {}	try {} catch {}	try {} catch {} finally {}
allowed	error (catch without try)	error	error	error

Nested try & catch is allowed

IMP

```
class C
{
static int m1(){
    try
    {
        try{}catch{}
    }
    catch{

    }
finally
    {
    }return 10;
}
```

~~Some code in other~~

Class

~~static~~ .

~~Exception test t = new Exception Test();~~

~~static~~

// int k = exceptionTest.m1();

Exception raise हो रहा है
और finally block return value
than this value will ~~se~~ override
the exception.

throws — used with method prototype, class name is used

throw — used in the method body with object

throws

eg ⇒ public void m1() throws AbcException;

throws kw is used with method prototype

throw kw is used in body of method

⇒ public void m2() throws Abc Exception
{
=
throw new XYZ Exception();
}

⇒ Throws indicate that when this method is call then keep it in try block;

eg m2(){
try {m1();
} catch(Exception e)
{
}
}

```
class ExceptionTest{
    static void m1(){
        int i=10/0; throw new ArithmeticException();
    }
    psv main(String[] args)
    {

    }
}
```

throw को बाद कुछ नहीं लिख सकते।
अगर किया तो // CTE

अगर compiler check कर रहा है की Exception handle हो रहा है या नहीं तो वो checked Exception है
otherwise unchecked Exception

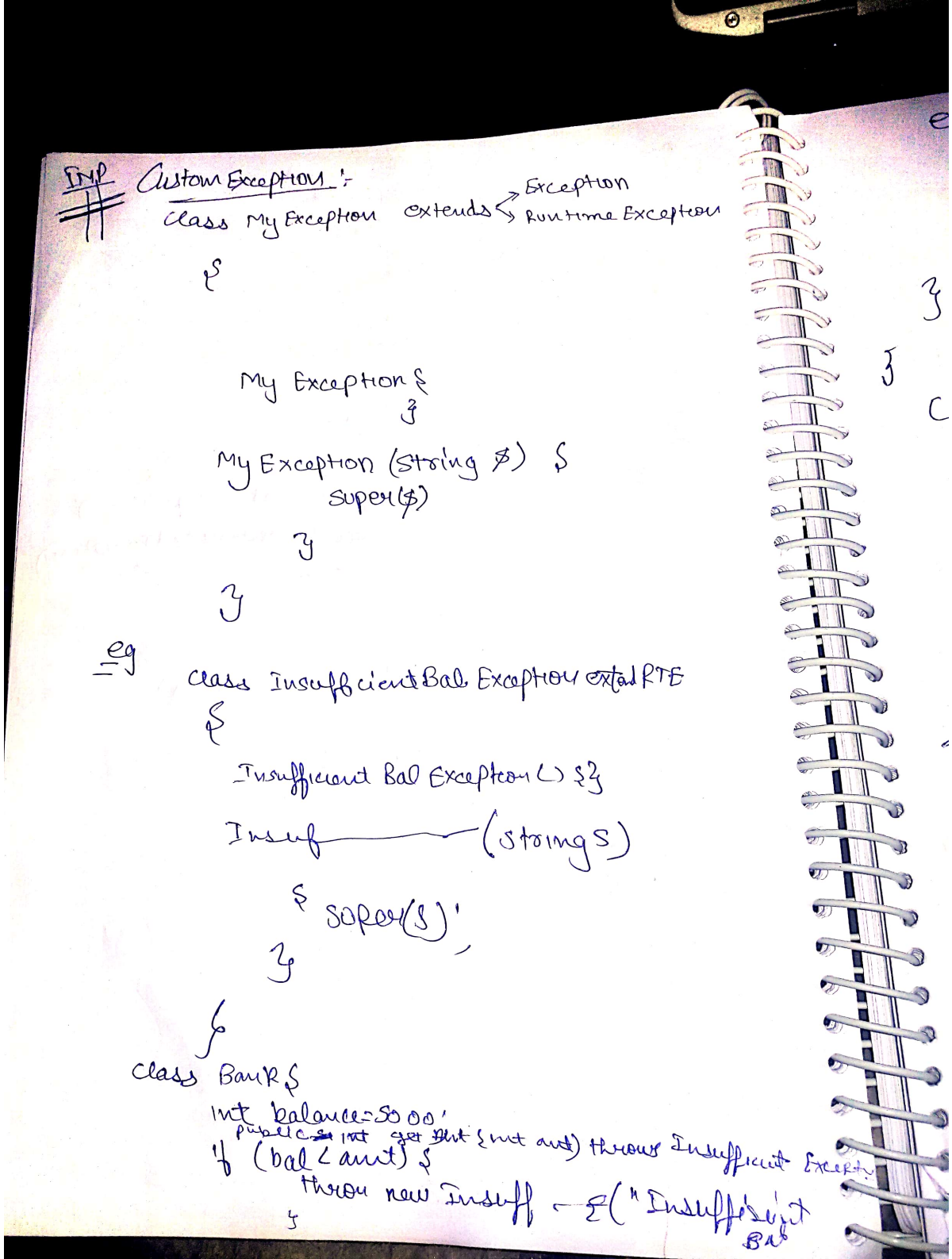
अगर checked Ex handle नहीं करते तो
CTE

```
class V
{
    static void m2 throws Exception
    {
        Thread.sleep(1000);
        SOP("m2");
    }
}
Public sv main (String[] args throws Exception
{
    SOP("main");
    m2();
}
}
```

Unchecked Exception को handle करेगा या ना करे।

In checked Exceptions

```
class E
{
    static void m1()
    {
        try {
            SOP("m111");
            Thread.sleep(1000);
        } catch(InterruptedException e)
        { }
    }
    PSV main(String[] args)
    {
        SOPm1();
    }
}
```


Imp Custom Exception :-

class MyException extends Exception / RuntimeException

{

}

MyException {

}

MyException (String $) {

super($)

}

}

eg

class InsufficientBalExcption extRTE

{

InsufficientBalException() {}

Insuf——(String s)

{ super(s);

}

}

class Bank {

int balance=5000;

public int getBal(int amt) throws InsufficientExcption

if (bal < amt) {

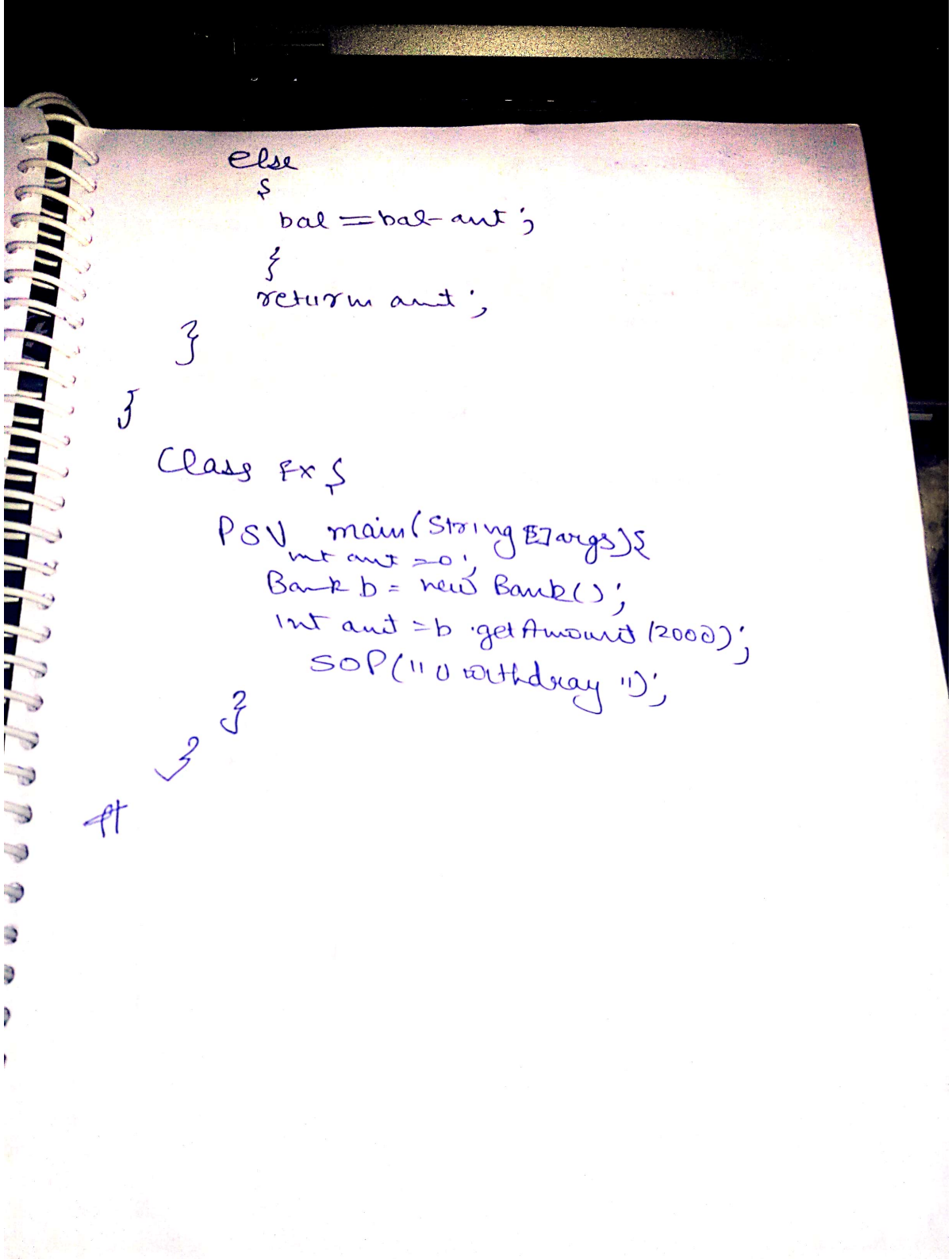
throw new Insuff——("Insufficient Bal

}

}

}

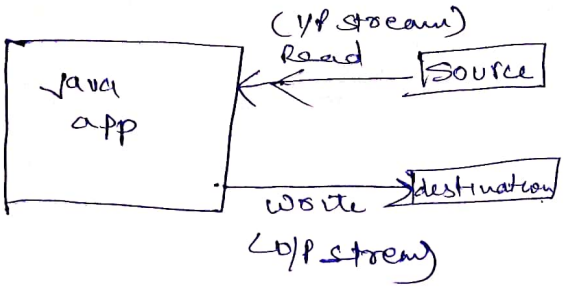
C



```
else
{
  bal = bal - amt;
}
return amt;
}
}
Class Ex {
  PSV main(String[] args){
    int amt = 0;
    Bank b = new Bank();
    int amt = b.getAmount(2000);
    SOP("U withdraw");
  }
}
```

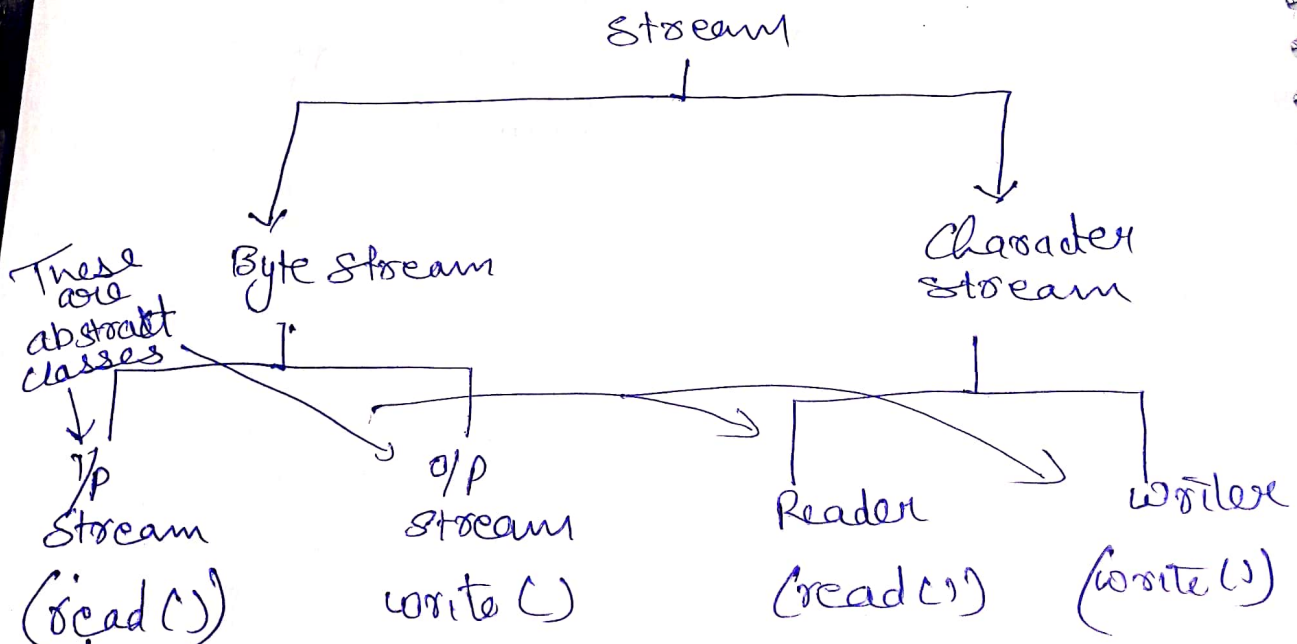
Streams :-

↳ flow of data



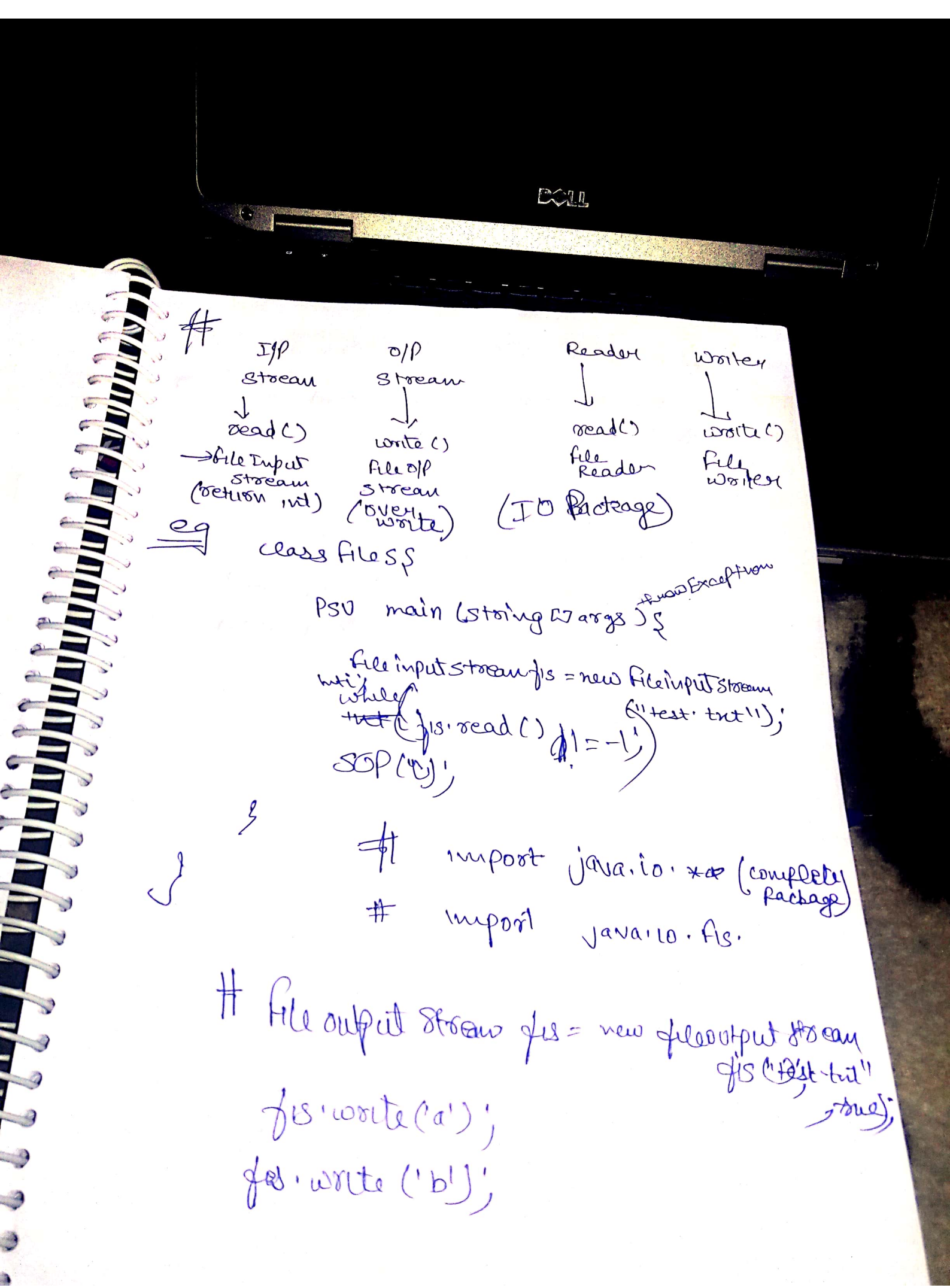
Type of Stream :-

↳ based on format of data



1 byte at a time

At the end it return -1



Disadvantage of I/P & O/P Stream is

object को remote area पर send नहीं कर सकते

It doesn't work with object

So

Object Input stream & Object Output Stream is used to save the state of object in file in the peces of byte is called serialisation

Java.io.Serializable

Eg Class student implements Serializable (Tag interface)

{

}

Eg

```
import java.io.*
class student implements Serializable {
    int rollno;
    int age;
    Student (int rollno, String name, int age)
    {
        this.rollno = rollno;
    }
}
class serialAndDeSerialTest
{
    PSV main(String[] args) throws Ex {
        Student s = new Student(11, "N", 35);
        FOS fos = new fileOS("stud.ser");
        ObjectOS oos = new ObjectOS(os);
        oos.writeObject(s)
        SOP("done");
    }
```