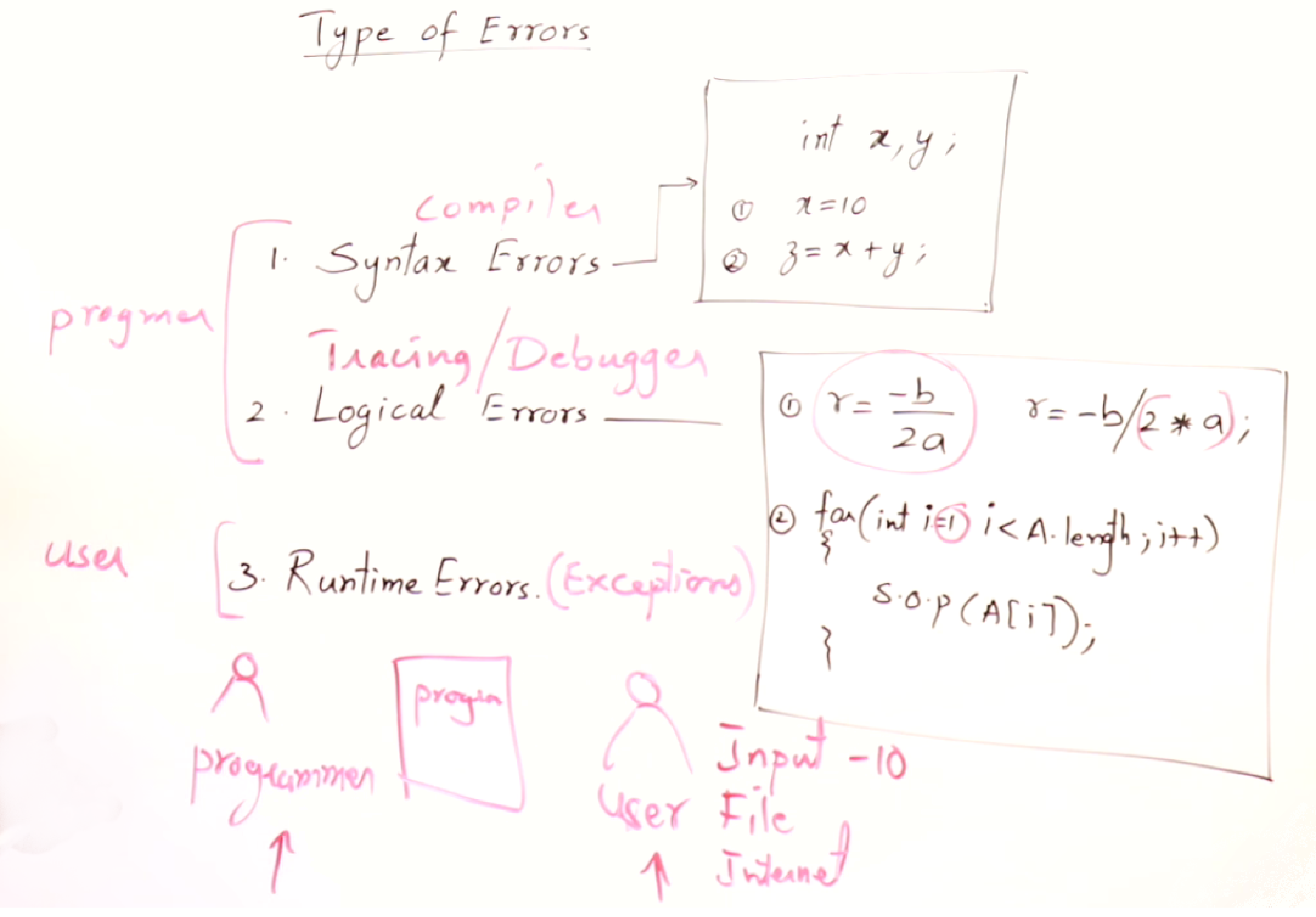


Type of Errors

programmer
1. Syntax Errors — compiler

```
int x, y;
① x=10
② z= x + y;
```

2. Logical Errors — Tracing / Debugger

```
① r = -b/2a      r = -b/(2*a);
② for(int i=1; i<A.length; i++)
  {
      s.o.p(A[i]);
  }
```

user
3. Runtime Errors. (Exceptions)

programmer
program
User
Input -10
File
Internet

--> Syntax and Logical errors are faced by Programmers, and runtime errors are faced by user.

--> Spelling or grammatical mistakes are syntax errors, for example using uninitialised variable it and using undefined variable etc and missing a semicolon etc. Syntax errors can be removed with the help of compiler.

--> Logical error is a bug in program that it to operate incorrectly, for example missing parenthesis in the calculation. Logical errors are removed with the help of debugger.

--> Mishandling of a program causes Runtime error. Due to bad input, unavailability of resources this error take place. Major problems with runtime errors is program will crash. Exception handling is process of responding to the runtime errors.

Exception Handling Construct

```
class Test
{
    p.s.v. main(..)
    {
        int a,b,c;
      try
      {
          a=10;
          b=2;
        → c=a/b;
          s.o.p("Result is "+c);
      }
      catch(ArithmeticException e)
      {
          s.o.p("Division by zero"+e);
      }
    }
}
```

--> Exception handling mostly ham "try......catch" block ka help sai karta hai.

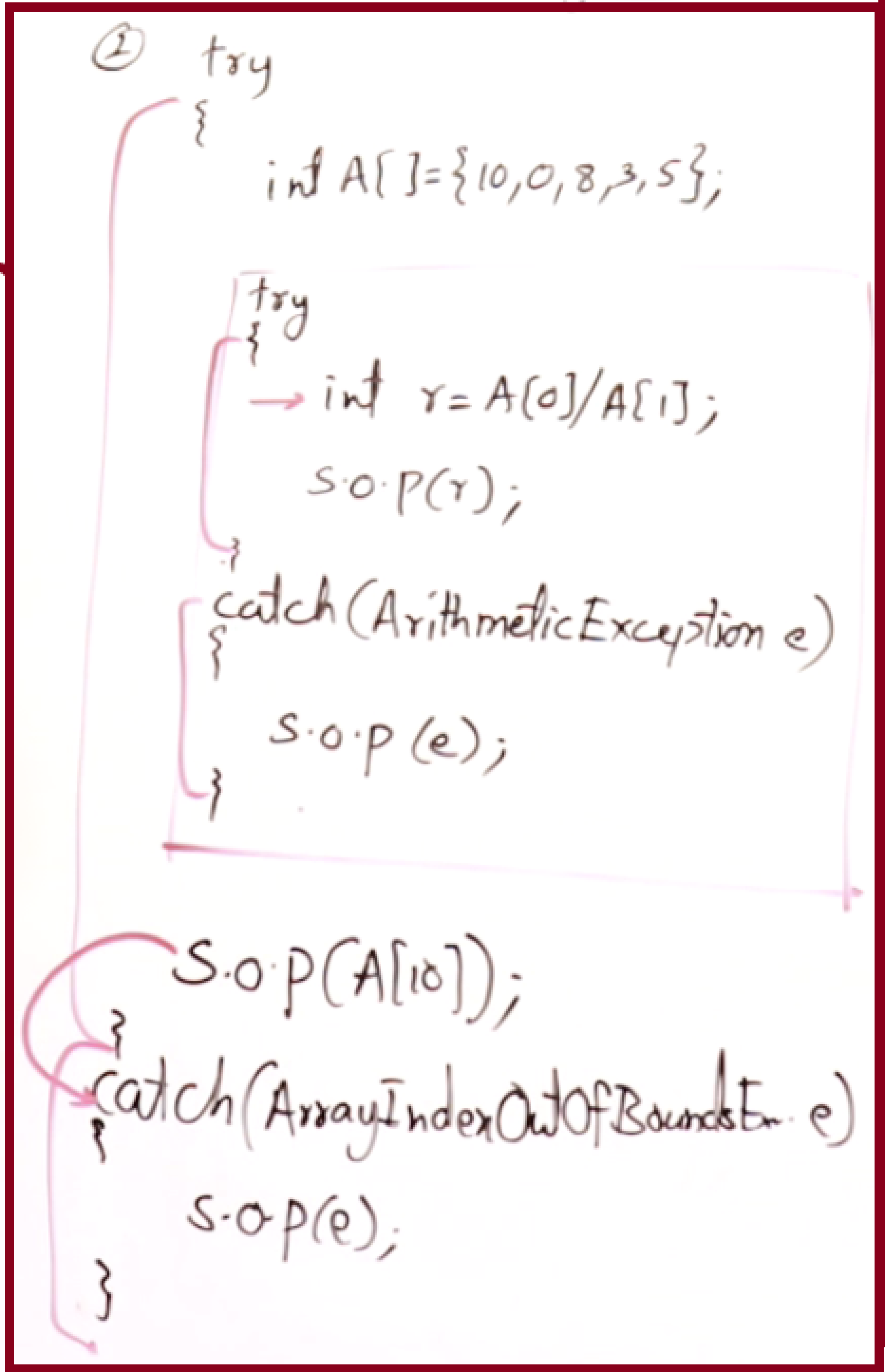
--> "try" mai ham wo code likhta hai jo hma during execution error dai sakta hai.

--> "catch" mai ham ise trah ka code likhta hai jisma user ko ya btata hai ki "try-block" mai kis trah ka error aa raha hai.

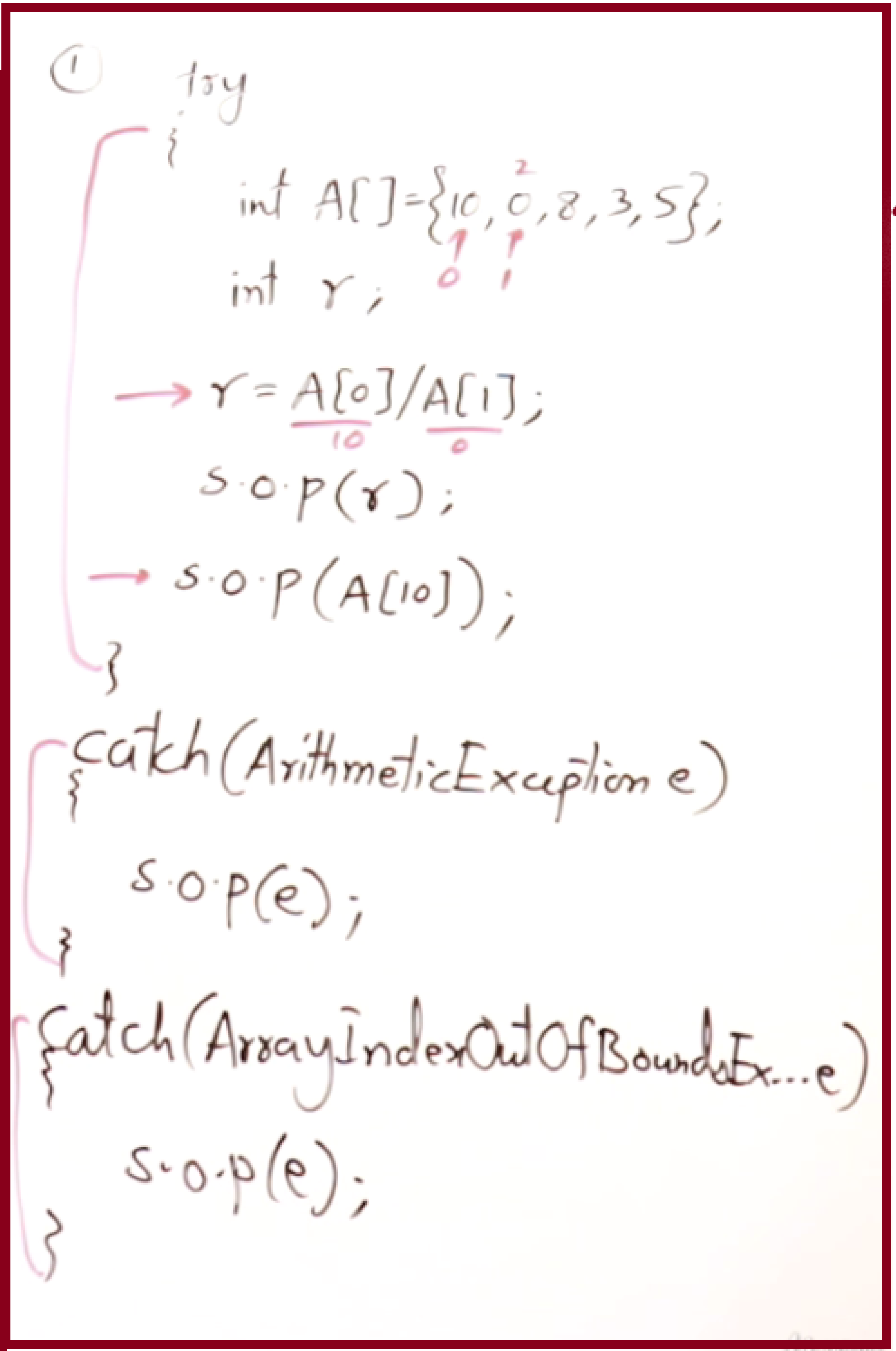
Udemy

Exception Handling Construct

Example of nested "try-catch block".

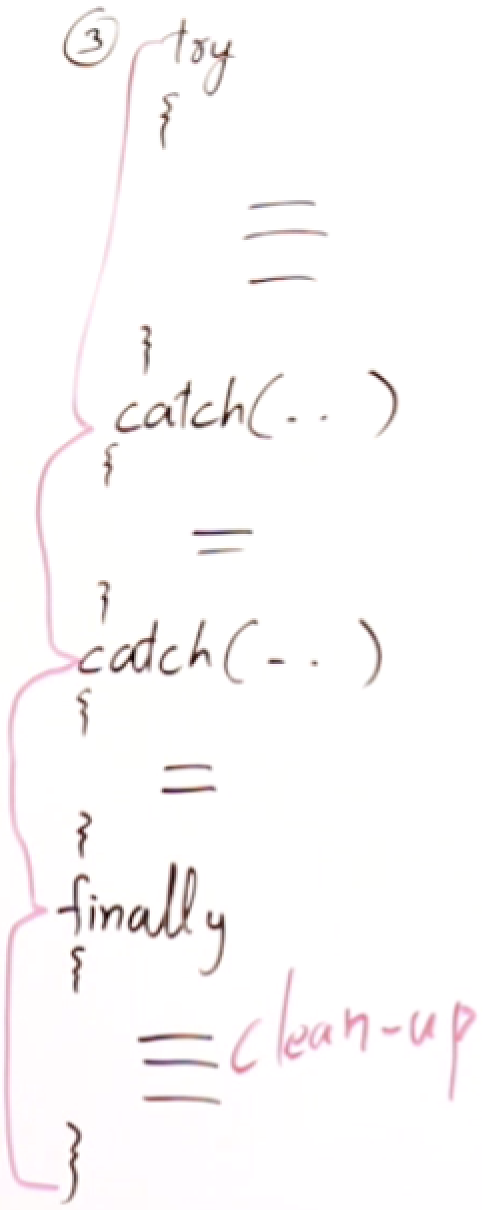


```
② try
   {
      int A[]={10,0,8,3,5};

      try
      {
      → int r=A[0]/A[1];
        s.o.p(r);
      }
      catch(ArithmeticException e)
      {
        s.o.p(e);
      }

      S.o.p(A[10]);
   }
   catch(ArrayIndexOutOfBoundsEx.. e)
   {
      s.o.p(e);
   }
```



```
① try
   {
      int A[]={10,0,8,3,5};
      int r;
   → r=A[0]/A[1];
      s.o.p(r);
   → s.o.p(A[10]);
   }
   catch(ArithmeticException e)
   {
      s.o.p(e);
   }
   catch(ArrayIndexOutOfBoundsEx...e)
   {
      s.o.p(e);
   }
```

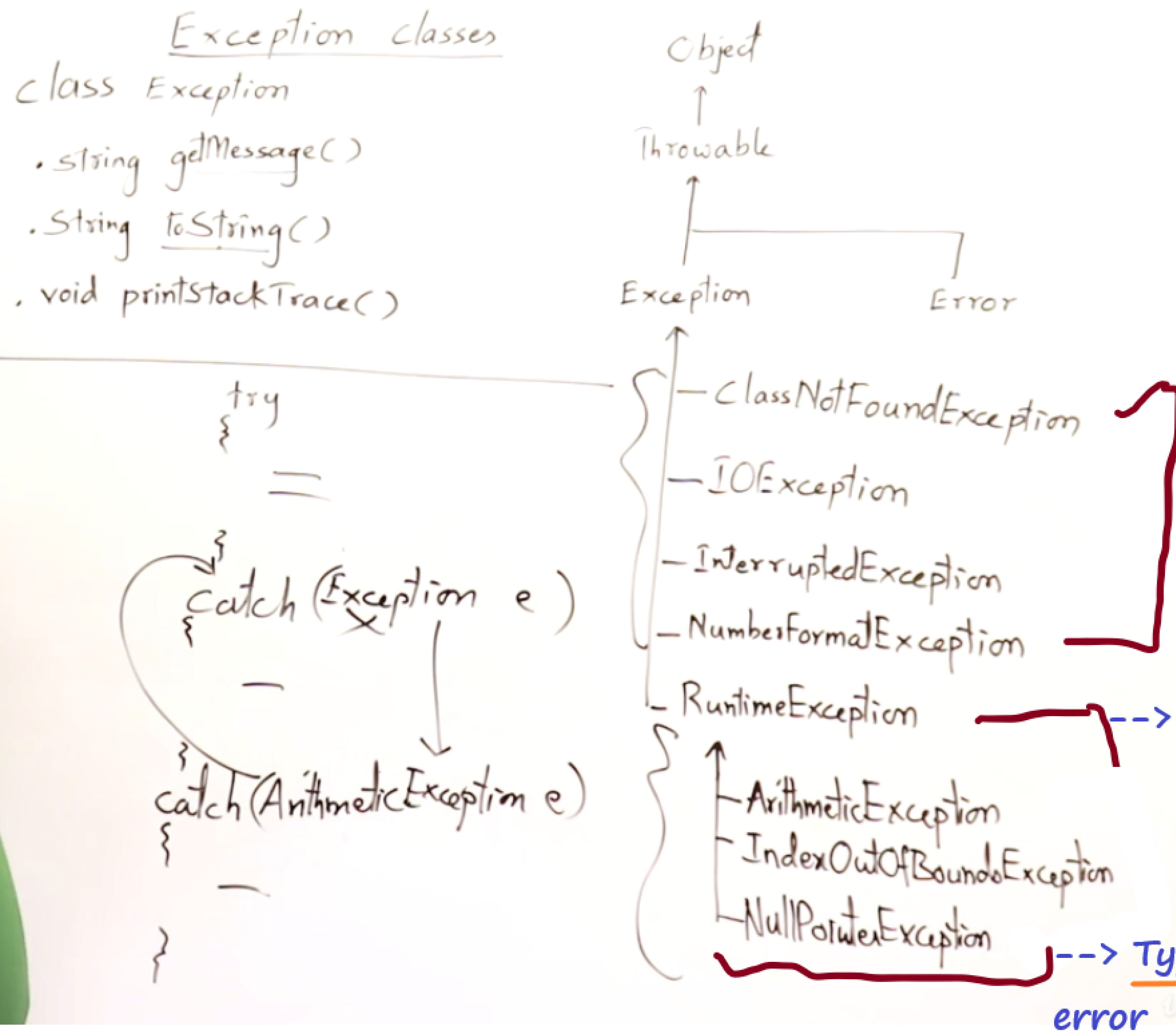
Example of multi-catch block.



--> "finally" ka use ham kisi bhi aisa code ko run ka liya karta hai jiska "try......catch" block ka baad chlna jaruri hota hai.

--> "try......catch" block ka result kya hoga usee sai "finally" block ko frak nahi partaa hai.

--> ya clean-up process mai bahut useful hota hai
(clean-up ka bara mai ham aaga padengai)



```
class Exception
 . String getMessage()
 . String toString()
 . void printStackTrace()

try
{
   =
}
catch (Exception e)
{
   —
}
catch (ArithmeticException e)
{
   —
}
```

⊙ checked error ko eliminate karna compulsory hota hai. Ise eliminate karna ka liya ham "try-catch" ka ise karta hai.

---> cHECKED ERROR

--> UNCHECKED ERROR

⊙ Unchecked error ko eliminate karna compulsory nahi hota hai.

--> Type of unchecked error

--> Exception ya par grand-parent class hai.
--> Checked and unchecked error parent class hai.
--> Type of unchecked error child class hai unchecked error ka liya.

Exception classes

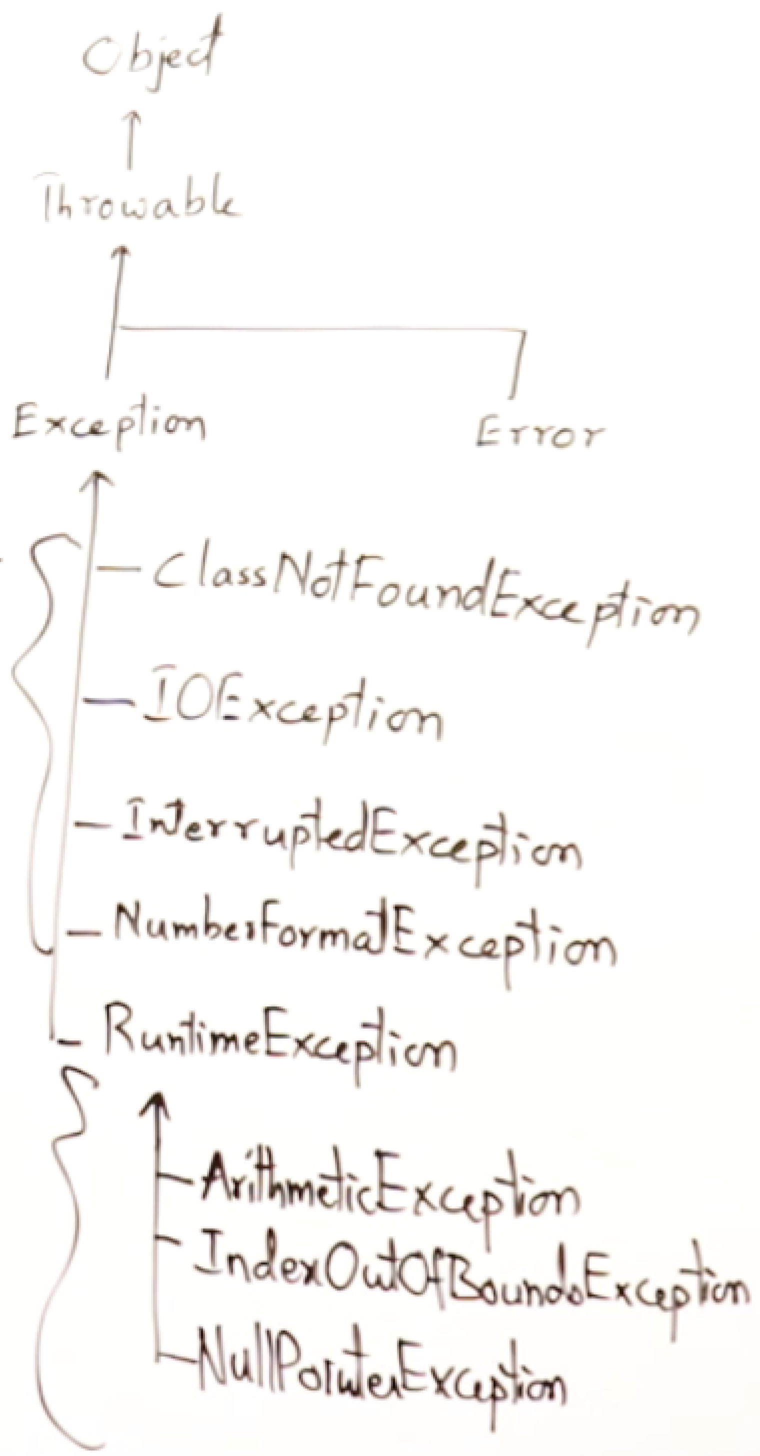
class Exception

. string getMessage()

. String toString()

. void printStackTrace()

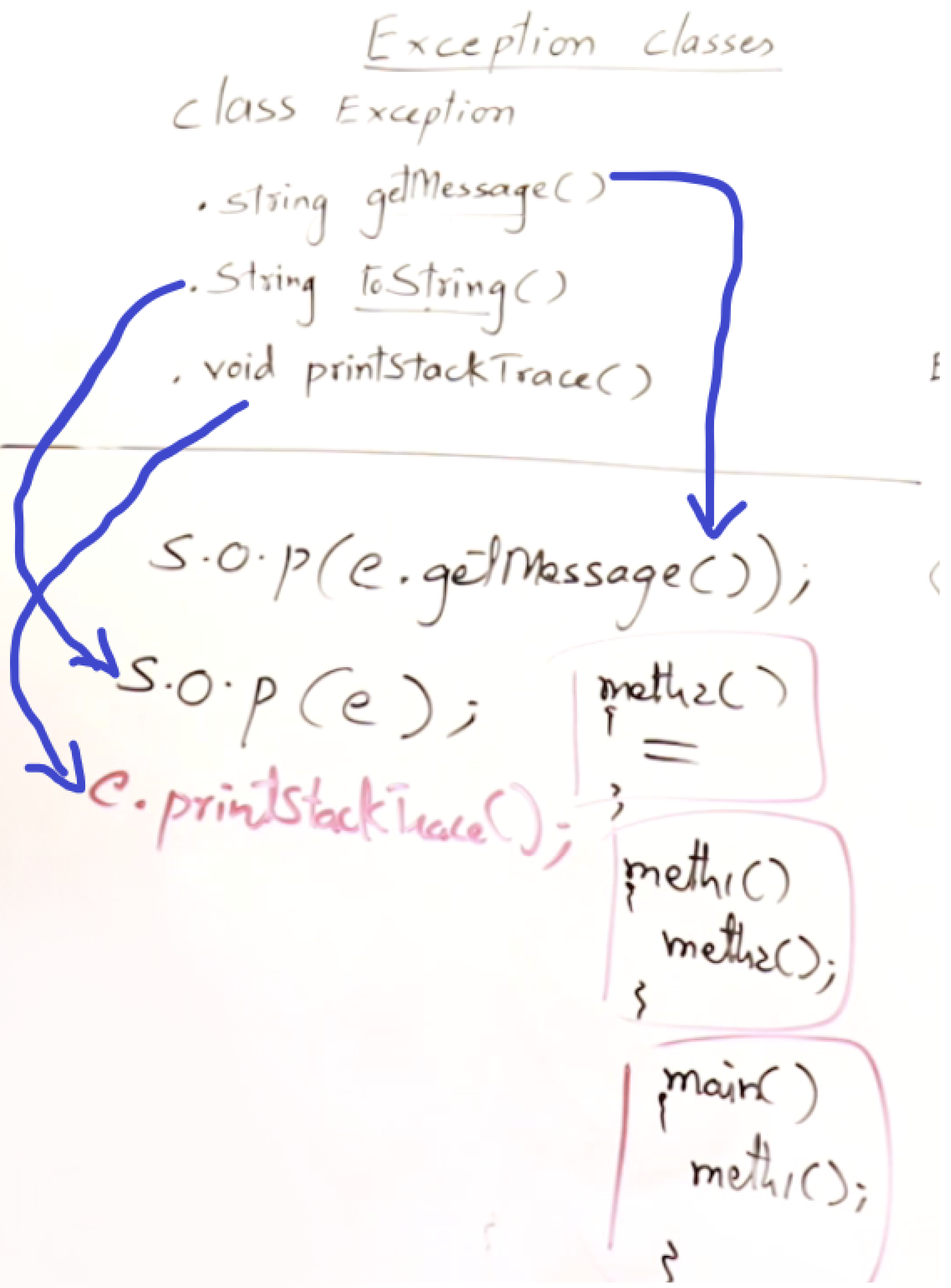
```
try
{
    =
}
catch (Exception e)
{
    —
}
catch(ArithmeticException e)
{
    —
}
```



--> Object is the mother class for all the java classes.

--> Exception is the parent class for all other exceptions.

--> jab bhi ham multiple or nested catch block likhta hai to starting block mai kabhi bhi super class sai start nahi karna chahiya.
Hmasa phela sub-class likhta hai. Kyuki aaga phela hi super class hoga then, wo sub class error ka liya check nahi karaga kyuki super class mai sabhi class already include hota hai.



--> "e" ko print karte hain, toh exception object ka toString() method ko print karta hain. Isme exception ki type, message, shaamil hoti hai. Toh aap console par kuch aisa dekhenge: "Error ----> java.lang.ArithmeticException: / by zero".

--> "e.getMessage()" specifically exception sai related error message ko show karta hai. Ye message normally hota hai aur exception kis wajah se hui usee galti ka bara mein batata hai. Agar ham ise ka use karta hai to hma terminal par aisa message show hota hai "/ by zero."

-->Normally, "e.getMessage()" ka istemal tab karenge jab aap user-friendly error message dikhana chahte hain, aur "e" ka istemal tab karenge jab aap debugging ya logging ke liye detail information show karna chahta hai

Exception classes

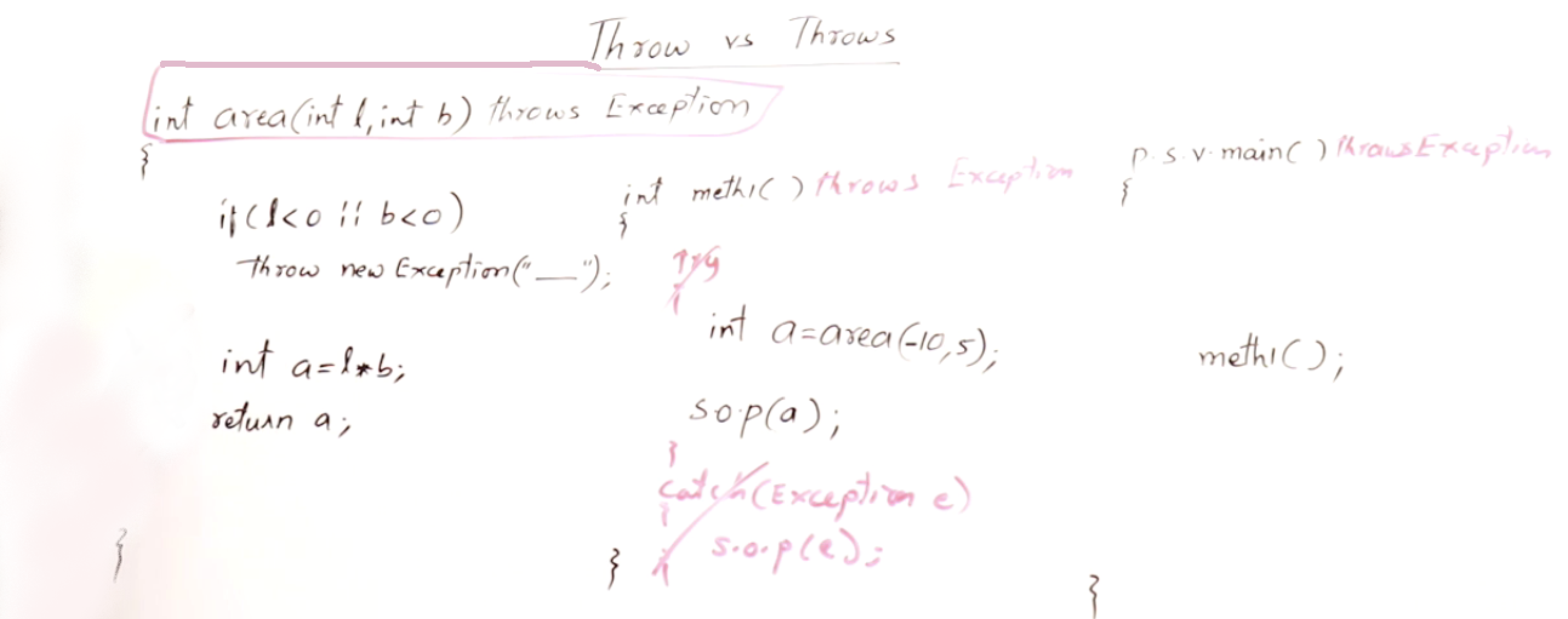
```
class Exception
  . String getMessage()
  . String toString()
  . void printStackTrace()
```

```
          minbalance = 5000

class MinBalanceException extends Exception
{
   public String toString()
   {
      return "minimum balance should be 5000,
                  try again with smaller amount";
}
```

--> Jab ham user defined exception class bnata hai to "extends Exception" likhna compulsory hota hai.

--> Ham bagal wala given class mai sai kisi ko bhi as method return type use kar sakta hai 1&2 mai sai.

--> Error ka message aapni marzi sai kuch bhi print karwa sakta hai.



Throw vs Throws

```
int area(int l, int b) throws Exception
{
    if(l<0 || b<0)
     throw new Exception("___");

    int a=l*b;
    return a;
}
```

```
int meth1() throws Exception
{
    try
    {
        int a=area(-10,5);
        sop(a);
    }
    catch(Exception e)
    {
        s.o.p(e);
    }
}
```

```
p.s.v.main() throws Exception
{
        meth1();
}
```

--> "Throw" ka use kar kai within method error find kar sakta hai.

--> We can check either checked or unchecked exceptions in java by "throw" keyword. The "throw" keyword is mainly used to show a custom exception.

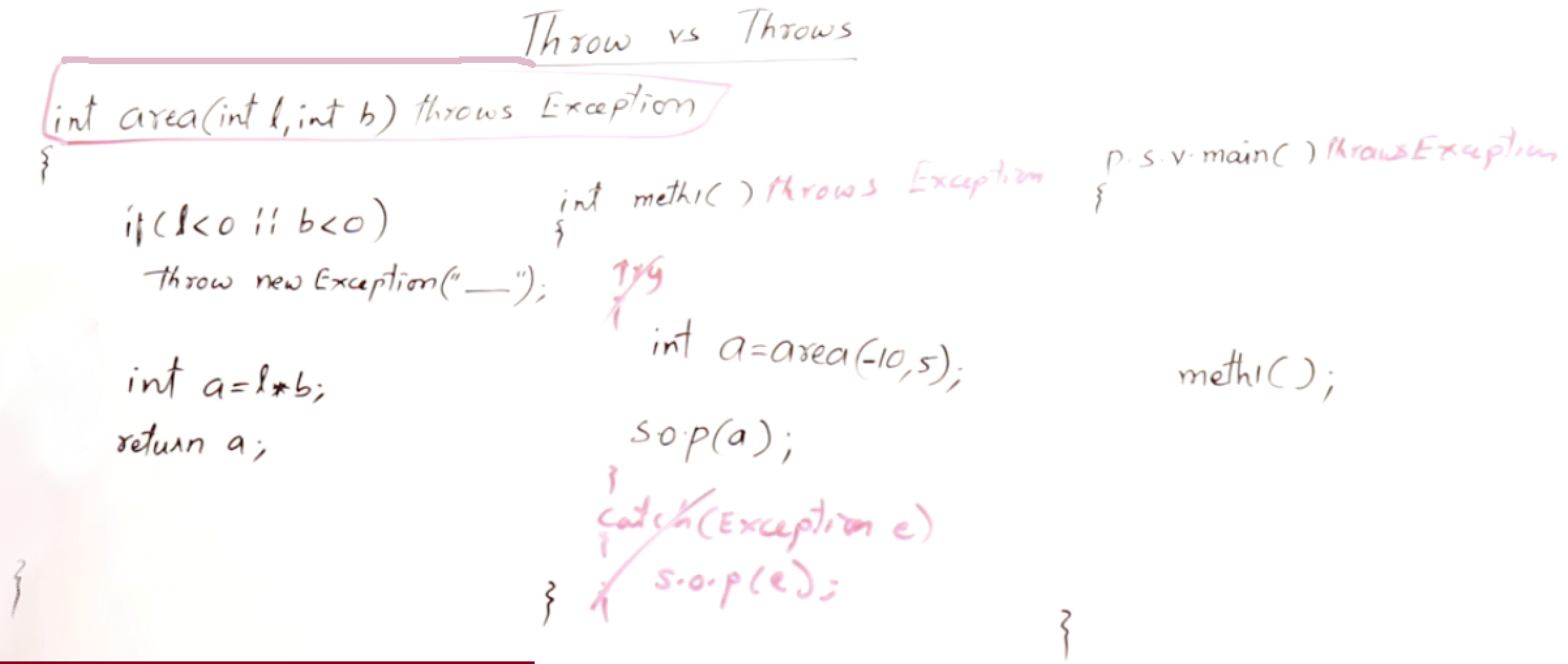
--> Program mai "Throw" keyword aata hi program wahi ruk jata hai and "catch" block ko khojta hai error handling ka liya.

--> Isko initialize karna ka liya aisa likhta hai:

Throw new Exception();

※ Bracket mai ham koi random message bhi likh sakta hai jo user ko show hoga.

※ ya "Exception" ki jga par koi aur naam bhi ho sakta hai aagar koi user defined Exception class hua to.

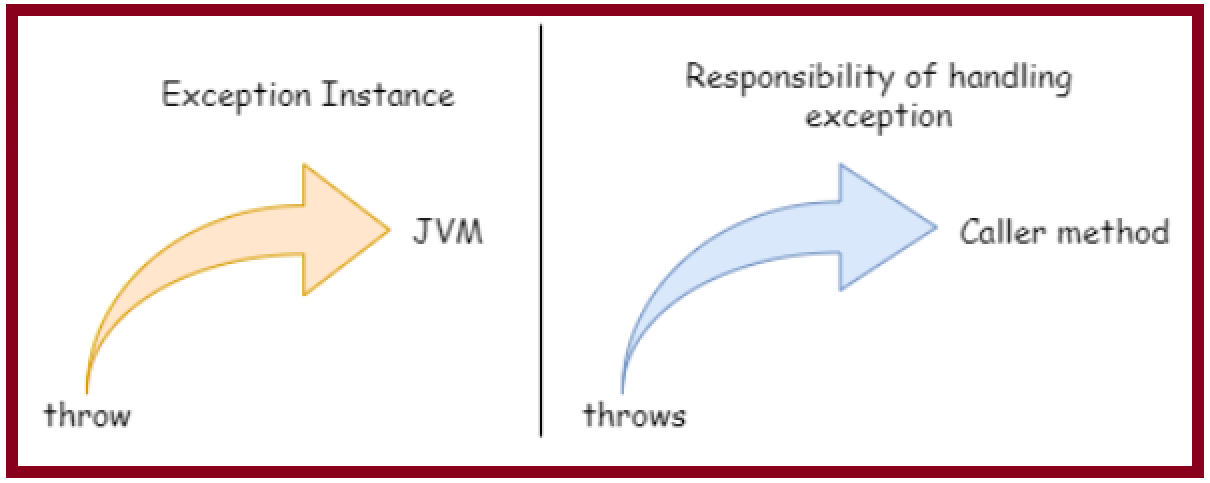


Throw vs Throws

```
int area(int l,int b) throws Exception
{
    if(l<0 || b<0)
      throw new Exception("___");

    int a=l*b;
    return a;

}

int meth1( ) throws Exception
{
  ~~try~~
  ~~{~~
      int a=area(-10,5);

      s.o.p(a);
  ~~}~~
  ~~catch(Exception e)~~
  ~~{~~
      ~~s.o.p(e);~~
  ~~}~~
}

p.s.v.main( ) throws Exception
{

      meth1( );

}
```

--> "throws" ka use kar ka haam caller method ko btata hai ki calling method mai koi exception ho sakta hai to aap ready rho error handling ka liya.

--> caller method mai exception handling ka liya "try....catch" hona jaruri hai nahi to error show hoga.

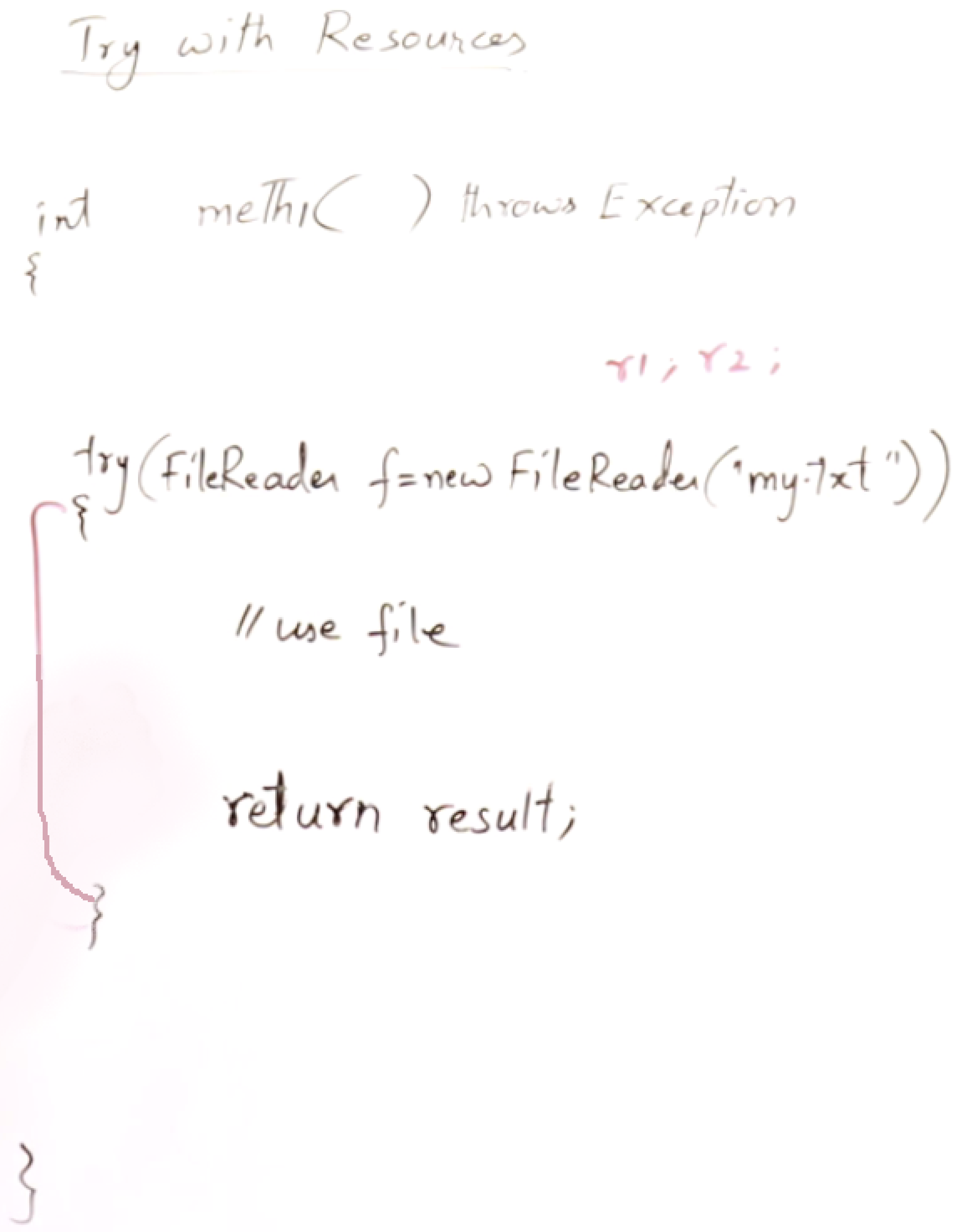


Throw vs Throws

```
class NegativeDimensionException extends Exception
{
    public void Sting toString( )
    {
        return "Dimension cannot be Negative";
    }
}
```

```
int area(int l, int b) throws NegativeDimensionException
{
    if(l<0 || b<0)
      throw new NegativeDimensionException();
    int a=l*b;
    return a;
}
```

ya par hmna custom Exception class bnaya hai custom Exception message show karna ka liya.



Try with Resources

```
int    methi(    ) throws Exception
{
                                   r1; r2;

  try(FileReader f=new FileReader("my.txt"))
  {
        // use file

        return result;
  }

}
```

--> "Try-with-resources" ek feature hai jo resource management ko aasaan banata hai. Iska use tab karta hai jab hum kisi resource jaise ki files, sockets, database connections, etc., ka use karta ha aur chahta ki wo automatically close ho jaya jab uska kaam khatam ho jaya.

--> Jab ham try block ke andar resources ko declare karte hain, to Java khud hi in resources ko try block ke bahar automatically close kar dega, chahe code execution normal tarike se ho ya fir exception aaye. Isse hma explicitly "close()" method ko call kar ka band nahi karna hota hai.

--> Ham ek hi try block ke andar multiple resources ko bhi declare kar sakte hain. In resources ko semicolon (;) se separate karna hota hai. Yeh resources ek saath close ho jayenge, aur unki close sequence reverse order me hoti hai.

Try with Resources

```
int    methi(   ) throws Exception
{
                              r1; r2;

  try(FileReader f=new FileReader("my.txt"))
  {
         // use file

         return result;
  }
}
```

※ Resources Initialization ka liya pehle ham ko resources ko initialize karna hota hai. Ham "try block" ka bracket ka andar resources ko declare aur initialize kar sakte hain.

※ Ise case mai "FileReader" ek resource hai jo file "my.txt" ko read karne ke liye use ho raha hai.

※ Try with resource mai ham use resource ko call kar sakta hai autoclosable hota hai.