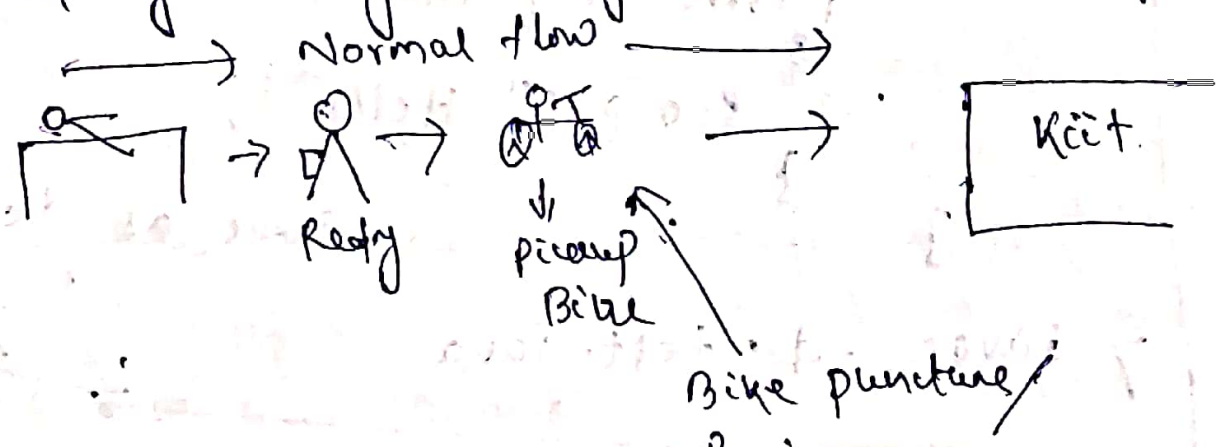
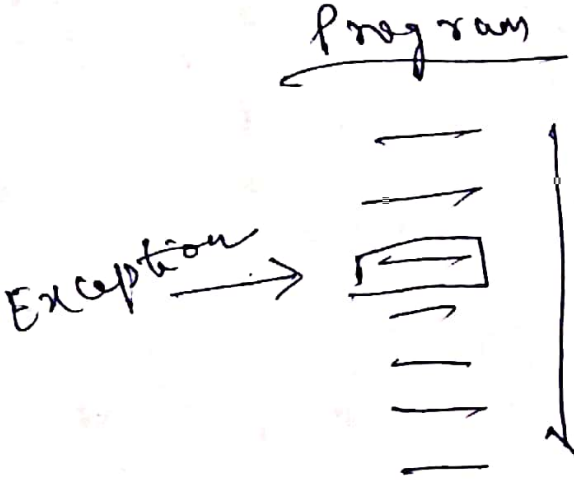
Exception handling.

Consider the Normal flow of your daily duty to go to the college.



*) This unexpected event will disturb your normal flow. This is called as exception. You may reach at office but is not in Right time.

def:- Exception is an unwanted or unexcepted event, which occurs during the execution of a program i.e. at Run time that disturbs the normal flow of the program.



How to handle it?

*) To find an alternative way so that the normal flow will not disturbs.
e.g. on case of going to college if there is a puncture we can take the help of your friend or OLA so that you can reach at college. This is called Exception Handling.

*) Now in case of a Program how to handle exception.

*) Advantages of Exception handling is it will not disturbs your normal flow of execution and the s/w will execute properly. s/w will not crash. Exception Handling is not to Repair or remove the error but finding an alternative way to execute the program.

Exception & Error

Q/ which is the Exception Hierarchy.

Q/ which is the parent class of all classes in java.
Ans - Object class.

'Throwable' is the parent class of all exception class.

Object (parent class)

↓

Throwable

Exception

1) (By our programs)

2) → Recoverable

3) → Two types

1) Compile time (checked)

2) Run time (unchecked)

Error

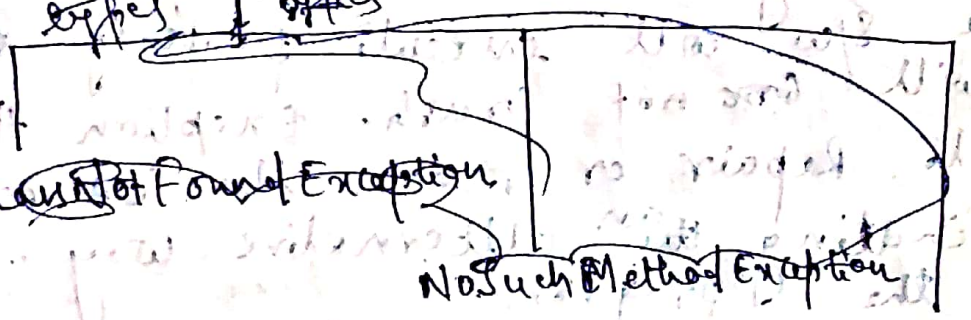
1) (Because of lack of system Resources)

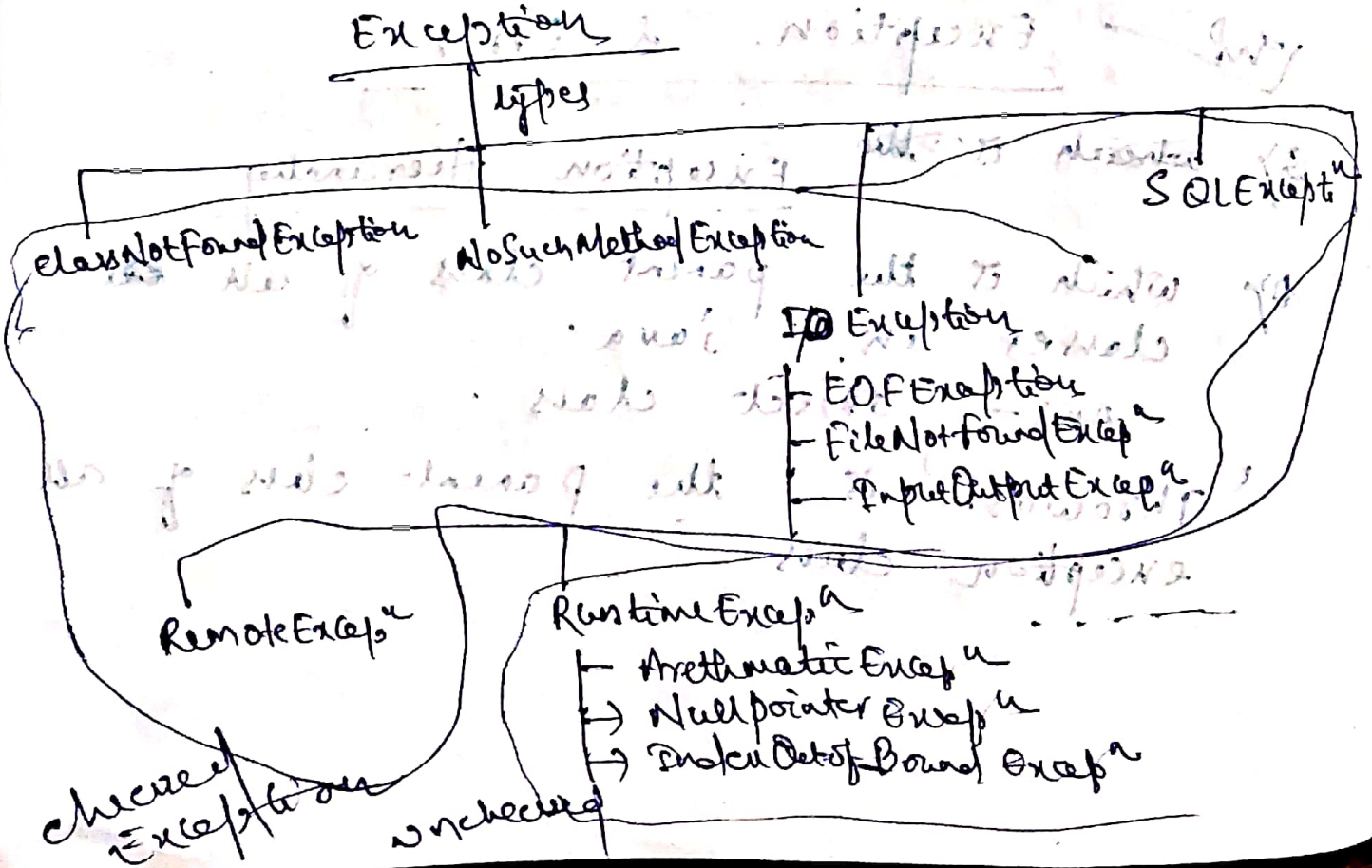
e.g. → less memory

→ slow processor.

2) → Not Recoverable

3) → only one type Run time (unchecked)





Checked and Unchecked Exception
or Compile & Runtime Exception.

*) Actually Exception are occured at Runtime. No Exception will occur at compile time.

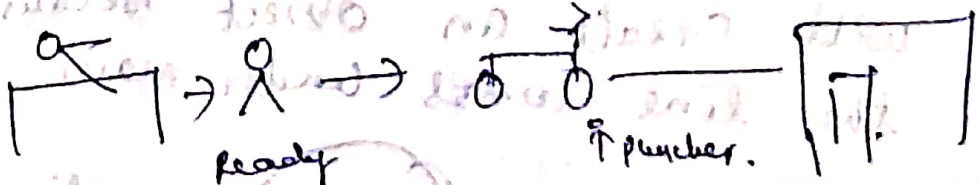
*) At compile time compiler will check whether in future there will be any exception will be occure or not : But ~~error~~ with exception actually occur at Runtime.

e.g.
```
import java.io.FileInputStream;
class test
{   psvm( )
    {   FileInputStream fis = new
            FileInputStream("d:/abc.txt");
    }
}
```

*) When we compile this it will give.
file not found Exception. must be caught or declared to be thrown.
i.e. compiler will warn to report this.

*) Run time exception is, compiler is not able to check i.e. unchecked.

e.g.



Mother compiler

checked
- → ProgNotFound
- → CompIDNotFound
- → LWCHNotFound Exception

Unchecked.
- → Bikepunctured Exception
- → Accident Exception.
- → Traffic Control Exception.

Handle Exception

*) First check & uncheck Exception than how to solve / Handle it.

e.g.: import java.io.FileInputStream;

class test

{ p.s.v.m (String [] args)

FileInputStream fis = new FileInputStream("d:/abc.txt");

↳ checked Exception

{ int a=100, b=0, c;

c = a/b;

S.o.p(c);

}

↳ Unchecked Exception.

How to handle:

try & catch

*) when c=a/b executes 'main' method will create an object because 'c=a/b' the line comes under main method

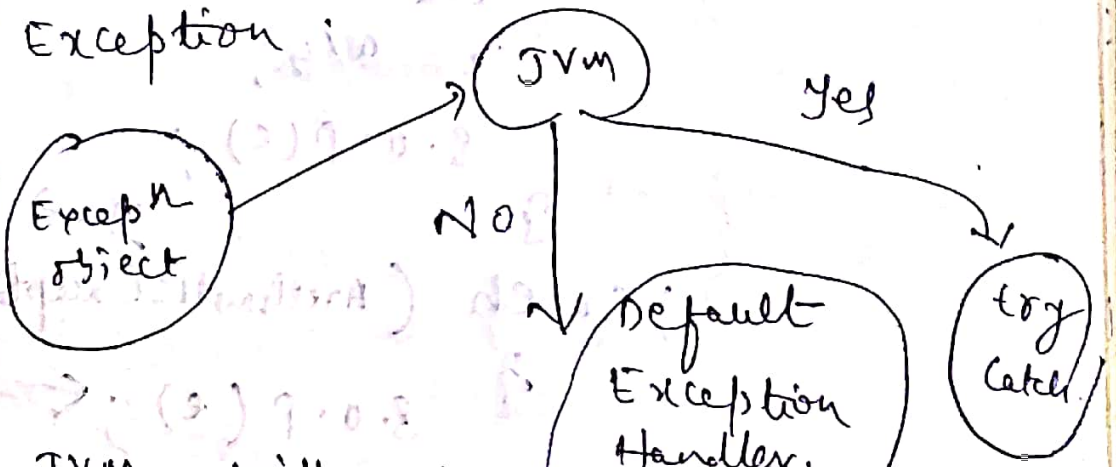
Exception object
→ excep^n name
→ excep^n Description
→ Stack trace (line).

*) Three things inside the object of created

→ Exception name
→ " Description
→ Stack Trace (line)

*) the main method will pass the object to JVM. If JVM will not handle the exception then JVM will pass it to the Default Exception Handler & terminate the program abnormally.

*) Default Exception Handler will print the Exception.



*) If JVM will Handle exception that means it will Handle through

try & Catch.

*) Exception Handling is done through Five key words.

1) try 2) Catch 3) finally 1) throw 5) throws

try Catch

Syntax

```
try
{
    //risk code
}
Catch (ExceptionclassName ref. var. name)
{
    //Handling code
}
```

e.g;

```
try
{
    int a=100, b=0, c;
    c= a/b;
    S.O.P(c);
}
Catch (ArethmeticException e)
{
    S.O.P(e);
}
```

or → Exception

Excep^n name // Descri^n Stack trace

Try / catch

Method to Print Exception.

```
class Test
{
  P.S.V.M(String args[])
  {
    try
    {
      int a=100, b=0, c;
      c = a/b;
      SOP(c);
    }
    catch(ArithmeticException e)
    {
      1) e.printStackTrace();
      2) S.O.P(e) or S.O.P(e.toString());
      3) S.O.P(e.getMessage());
    }
```

→ Except Name
→ Descriptn
→ Stack trace

1) e.printStackTrace() → Name, Descriptn, line no)

2) SOP(e), SOP(e.toString()); → Name, Descriptn
✗ printStackTrace

3) SOP(e.getMessage)
→ ✗ Name
→ Descriptn
✗ → stackTrace.

1) e.printStackTrace()
2) → S.O.P(e) or S.O.P(e.toString());
3) → S.O.P(e.getMessage());

Finally.

*) finally is the block that is always executed whether exception is handled or not.

Syntax:-

```
try
{
}
catch (Exception e)
{
}
finally
{
}
```

1)

```
try
{
}
finally
{
}
```

II)

*) If exception occurs inside try block then control goes to catch then goes to finally.

*) If exception doesn't occur inside try block then control doesn't go to catch block & directly goes to finally block.

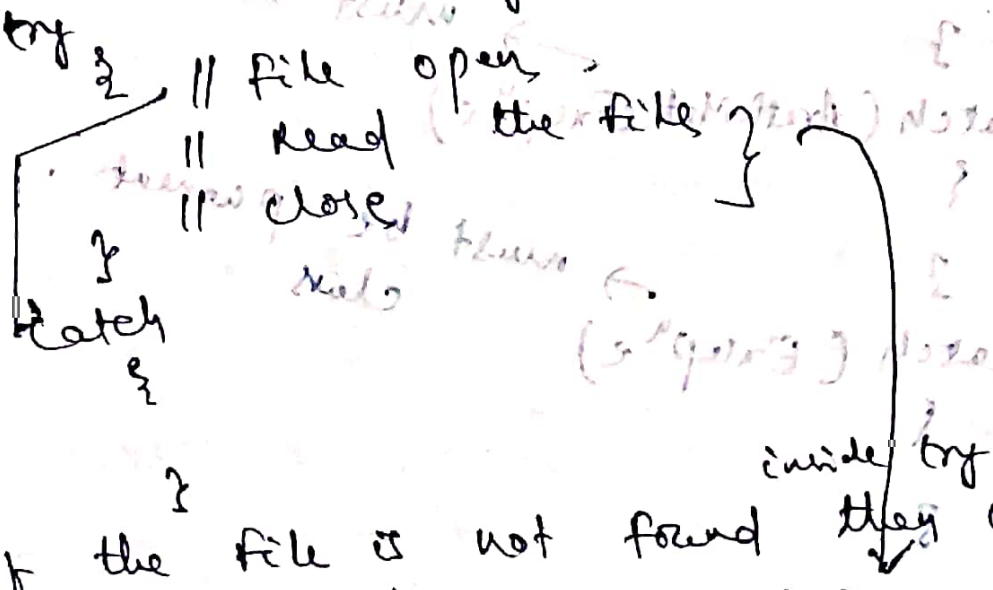
*) Finally block always follows either try block or by catch block

*) finally block is used to clean up code.

e.g. Database close code
File close code
Memory Release code.

*) The Resource which are opened inside try block are closed at finally block.

*) If we close any code inside try block but not in finally then.



If the file is not found then control goes to catch block & skiping Read file and close file: so file will not closed.

Difference between final, finally & finalize

final	finally	finalize.
1) Keyword	1) Block	1) Method
2) Use with variable, method & class	2) Use with either try or try catch block	2) method is overrride for any object
→ final variable (i.e. value is constant)	try { } catch { } finally { }	try { } finally { }
→ final method (does not override)		
→ final class (class is not inherited)		

Scanned with OKEN Scanner

Various combination of try, catch & finally.

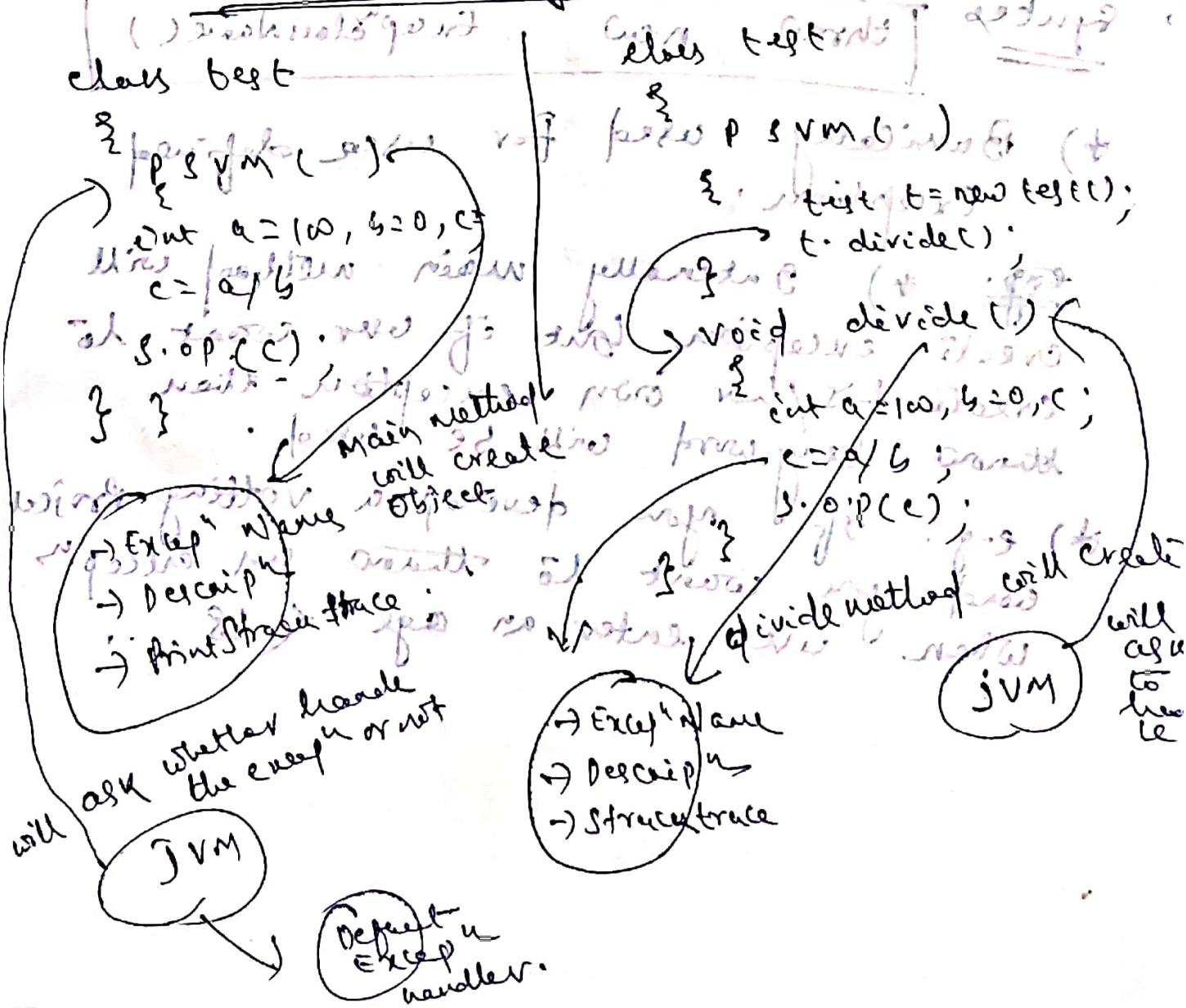
I)
```
try{
}
catch(ArithMeticExcep e)
{
}
catch(Excep e)
{
}
```
→ must be child
→ must be parent class

II)
```
try{
}
catch(ArithmeticExcep e)
{
}
catch(ArrayIndexOutOfBoundsException e)
{
}
```

III)
```
try{
    try{
    }
    catch(){
    }
catch(){
}
```

iv) try{ } try{ }
catch( ) { } catch( )
try{ } finally( )
catch( ) { }

Throw Keyword :



*) JVM will ask to divide method whether he will handle the excep^n or not. If not then JVM terminated

*) If not then JVM will ask to caller method i.e. to divide(). Whether he will handle it or not. If not JVM will terminate main method & pass the object to default Exception handler.

*) To handle the Excep^n we will use try-catch block.

Syntax: `throw new Exep$^n$Classname()`

*) Basically used for user defined Exception.

*) Internally main method will create exception. But if user want to create his/her own exception then throw keyword will be used.

*) e.g. If you develop a Voting Project and you want to throw an excep^n when user enter an age < 18.

```
class AgeException extends RuntimeException
{
    AgeException(String msg)
    {
        super(msg);
    }
}
```

```
class Voting
{
    p.s.v.m ( )
    {
        int age = 16;
        if (age < 18)
        {
            throw new AgeException
            (" you are not eligible");
        }
        else
        {
            S.o.p(" plz vote");
        }
    }
```

Scanned with OKEN Scanner

throws

*) "throws" key word is used to declair an exception. It gives an informat to the caller method that there may occur an excep^n so to provide the ~~error~~ excep^n handling code.

e.g.

You

```
import java.util.fileinputStream;
class ReadFile
{
void readfile()
{
fileinputStream fis = new fileInputStream
                      ("d:\asc.txt");
```

=

```
}
}
```

Exception

It will throw fileNotFoundException

*) Now if your friend will Access your program to his program He will handle the exception.

Scanned with OKEN Scanner

*) This exception is checked/compile time Exception.

*) throws key word will use with the method or inside the method.

e.g. void readfile() throws FileNotFoundExcep^n

*) throws keyword will indicate the caller method (i.e. the method which calles void readfile() method) that FileNotfoundException you have to handle in your program.

*) It is just an Indication to caller method

Your friend

```
class test
{
    p.s.v.m(  )
    {
        ReadWrite rw = new ReadWrite();
        try
        {
            rw.readfile();
        }
        catch (FileNotFoundException e)
        {
            s.o.p(e);
        }
    }
}
```

indication

Difference between throw & throws

throw	throws
1) create an exception object manually Programmer is responsible to create an exception object.	1) Indication to caller method that the exception may occur. So handle the exception.
2) Used for Runtime Exception / Unchecked Exception	2) Used for compile time Excep^n / checked Exception.
3) Only one exception can be used	3) Multiple Excep^n can be used.
4) Used inside the method	4) Used with the method
5) followed by "New" key word. followed by object / instance	5) Not followed by "new". followed by class.