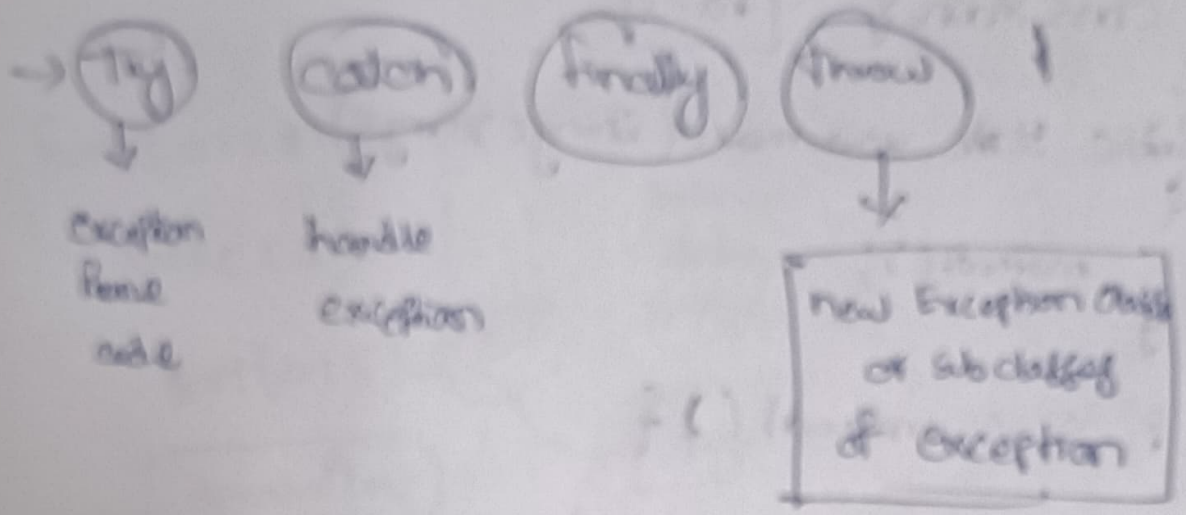
5 Keywords



→ throws → used with method to tell that method might throw an exception

(1) Try

→ used to specify code which can throw exception

→ Try is followed by either Catch (or) finally

eg:-

```
try{
    // error Prone Code
}catch(Exception e){
    // Solution or alternate
       code
}
```

```
try{
    // error Prone code
}finally{

}
```

---

(2) Catch

→ Catch is used to Provide alternate solution to exception occured in try block.

→ multiple catch block can be used.

eg:-

```
try{

}catch(__){

}catch(__){

}
```

good- example

```
Public class Main {
    Public S V M ( S [] args ) {
        try {
            method1 ("dummy");
        }
        Catch (ClassNotFoundException e)
        {
            // handle it
        }
        Catch (InterruptedException e)
        {
            // handle it
        }
        Catch (FileNotFoundException e)
        {

        }

P S V method1 (String name) throws ClassNotFoundException, InterruptedException {
    if (name.equals ("dummy"))
        Throw new ClassNotFoundException();
    else
        Throw new InterruptedException();
}
```

Error → Catch block only catch block thrown by try block

edge-case

```
try {

}
catch (SpecificException e)
{
}

catch (Exception e){

}
```

Specific

Generic

✓

```
try {

}
Catch (Exception e)
{
}

Catch (ClassNotFound e)
{
}
```

Generic

Specific

✗

Catch- multiple exceptions in one block

Catch.

```
try {
   method("dummy");
}
Catch (ClassNotFoundException | InterruptedException e)
{  if ( e instanceof ClassNotFoundException) // handle
   else if ( e instanceof InterruptedException) // handle
}
catch ( Exception e)
{

}
```

(2.) Try - catch - finally (or) try - finally

(i) try
catch
finally

(ii) try
finally

(ii) Finally block will "always get" executed, even if you "return" from try block or catch block

(iii) Only "one" finally block is allowed

(iv) Mostly used for closing objects (or) logging

```
open()
close()  } files
```

(v) If "JVM related issues" like out of memory, System Shutdown or Process is forcefully killed the finally blocks donot get executed

(vi) If we are using just

```
try {
    method1();  -> throws Checked exception
}                  Compile time exception
finally {

}
```

Then mandatory that the method calling method1() to use throws keyword

x

if its "unchecked exception / RTE" no need of throws keyword

Throw

-> use to throw a new exception

-> TO "Re-throw" the exception

```
class
{
  psvm(String[] args) throws ClassNotFoundException {
    try{
      method1();
    }
    catch(ClassNotFoundException e)
    {
      // DO Logging then throw
      throw e  // throws exception to caller, he should handle it in this case caller of main is jvm jvm should handle it
    }
}
```

```
ps v method1() throws ClassNotFoundException{

throw new ClassNotFound()

}
```

Creating Custom Exception

or child classes of exception

```
Class MyCustomException extends Exception {

    MyCustomException (String message)
    {
        super(message);
    }

}
Public class Main {

    PSVm(String[] args){
    try{
      method1 ();
    }
    catch (MyCustomException e )
    {
      // handle it
    }
    }

    PSV method1 ( ) throws MyCustomException {

        throw new MyCustomException ("exception occured");
    }
```

Why exception Handling? | Advantages

-> Code clean by Separating error handling code from regular code
-> allows Program to recover from error
-> allow us to add more info, helps in debugging
-> ~~Improve Security by hiding Sensitive information~~

Code Snippets refer github

---

disadvantages

-> its little expensive, if Stack trace is huge and its not handled or handled @ Parent Class