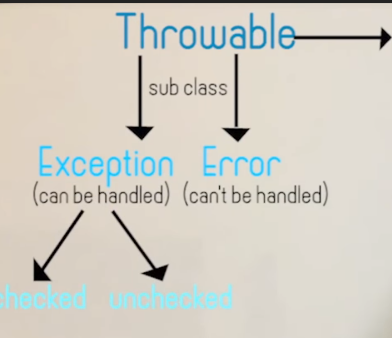
Throwable is a class:



Under checked exception,we have IO Exception and SQL Excepton

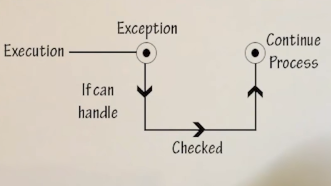
Under Unchecked exceptions ,we have run time exception

Statement is of two types:

* Normal->int i=1;
* Critical->int k=i/j;

Try block:

It will create object of exception and send to catch block



Throws-simply throws the exception above the stack

Arithmetic exception –it extends arithmentic exception-it extends exception

**public** **class** DemoException {

**public** **static** **void** main(String args[])

{

**int** i,j,k;

i=5;

j=0;

**try** {

k=i/j;

} **catch** (ArithmeticException e) {

// **TODO** Auto-generated catch block

System.***err***.println(e);

}

}

}

Output:

java.lang.ArithmeticException: / by zero

Array out of bounds exception:

**public** **class** DemoException {

**public** **static** **void** main(String args[])

{

**int** i,j,k;

i=5;

j=0;

**int** a[]=**new** **int**[4];

**try** {

**for** (**int** l = 0; l <=4; l++) {

a[l]=l+1;

}

**for** (**int** l : a) {

System.***err***.println(l);

}

}

**catch** (ArithmeticException e) {

// **TODO** Auto-generated catch block

System.***err***.println(e);

}

**catch** (ArrayIndexOutOfBoundsException e) {

// **TODO**: handle exception

System.***out***.println(e);

}

}

}

Always write catch Exception in the last

**catch** (Exception e) {

// **TODO**: handle exception

}

Checked exception:

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**public** **class** DemoException {

**public** **static** **void** main(String args[])

{

**int** i,j,k;

i=5;

BufferedReader bt=**new** BufferedReader(**new** InputStreamReader(System.***in***));

**try** {

j=Integer.*parseInt*(bt.readLine());//checked exception

**int** a[]=**new** **int**[4];

k=i/j;

**for** (**int** l = 0; l <=4; l++) {

a[l]=l+1;

}

**for** (**int** l : a) {

System.***err***.println(l);

}

}

**catch** (ArithmeticException e) {

// **TODO** Auto-generated catch block

System.***err***.println(e);

}

**catch** (ArrayIndexOutOfBoundsException e) {

// **TODO**: handle exception

System.***out***.println(e);

}

**catch** (IOException e) { //checked exception

// **TODO**: handle exception

}

}

}

Finally:

It executed the statement at the end of program

Throws –keyword in java to supress error

Throw- to give appropriately the error based on condition

**public** **static** **void** main(String args[]) **throws** Exception //throws implemntation

{

**int** i,j,k;

i=5;

BufferedReader bt=**new** BufferedReader(**new** InputStreamReader(System.***in***));

System.***out***.println("Enter");

j=Integer.*parseInt*(bt.readLine());//checked exception

**if**(j<10)

**throw** **new** ArithmeticException(); //throw implementation

**int** a[]=**new** **int**[4];

k=i/j;

**for** (**int** l = 0; l <=4; l++) {

a[l]=l+1;

}

**for** (**int** l : a) {

System.***err***.println(l);

}

}

User defined exception

**public** **class** trywithresi {

**public** **static** **void** main(String args[])

{

**int** i;

i=5;

**try** {

**if**(i<10)

**throw** **new** MyException("erreo");

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

System.***out***.println(e);

}

}}

**class** MyException **extends** Exception

{

**public** MyException(String Message)

{

**super**(Message); //to call the super calss sxception

}

}

The **try** statement can be nested. That is, a **try** statement can be inside the block of another **try**.

Each time a **try** statement is entered, the context of that exception is pushed on the stack. If an

inner **try** statement does not have a **catch** handler for a particular exception, the stack is

unwound and the next **try** statement’s **catch** handlers are inspected for a match. This continues

until one of the **catch** statements succeeds, or until all of the nested **try** statements are exhausted.

If no **catch** statement matches, then the Java run-time system will handle the exception.