**try and catch block**

**try block**

Inside **try block** we write the block of statements which causes executions at run time in other words try block always contains problematic statements.

**Important points about try block**

* If any exception occurs in try block then CPU controls comes out to the try block and executes appropriate catch block.
* After executing appropriate catch block, even through we use run time statement, CPU control never goes to try block to execute the rest of the statements.
* Each and every try block must be immediately followed by catch block that is no intermediate statements are allowed between try and catch block.

**Syntax**

**try**

{

.....

}

/\* Here no other statements are allowed

between try and catch block \*/

**catch**()

{

....

}

* Each and every try block must contains at least one catch block. But it is highly recommended to write multiple catch blocks for generating multiple user friendly error messages.
* One try block can contains another try block that is nested or inner try block can be possible.

**Syntax**

**try**

{

.......

**try**

{

.......

}

}

**catch block**

Inside **catch** block we write the block of statements which will generates user friendly error messages.

**catch block important points**

* Catch block will execute exception occurs in try block.
* You can write multiple catch blocks for generating multiple user friendly error messages to make your application strong. You can see below example.
* At a time only one catch block will execute out of multiple catch blocks.
* in catch block you declare an object of sub class and it will be internally referenced by JVM.

**Example without Exception Handling**

**Example**

**class** ExceptionDemo

{

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

{

**int** a=10, ans=0;

ans=a/0;

System.**out**.println("Denominator not be zero");

}

}

Abnormally terminate program and give a message like below, this error message is not understandable by user so we convert this error message into user friendly error message, like "denominator not be zero".

**Example of Exception Handling**

**Example**

**class** ExceptionDemo

{

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

{

**int** a=10, ans=0;

**try**

{

ans=a/0;

}

**catch** (Exception e)

{

System.**out**.println("Denominator not be zero");

}

}

}

**Output**

Denominator not be zero

**Multiple catch block**

You can write multiple catch blocks for generating multiple user friendly error messages to make your application strong. You can see below example.

**Example**

**import** java.util.\*;

**class** ExceptionDemo

{

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

{

**int** a, b, ans=0;

Scanner s=**new** Scanner(System.**in**);

System.**out**.println("Enter any two numbers: ");

**try**

{

a=s.nextInt();

b=s.nextInt();

ans=a/b;

System.**out**.println("Result: "+ans);

}

**catch**(ArithmeticException ae)

{

System.**out**.println("Denominator not be zero");

}

**catch**(Exception e)

{

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

}

}

}

**Output**

Enter any two number: 5 0

Denominator not be zero

**finally Block in Exception Handling**

Inside **finally**block we write the block of statements which will relinquish (released or close or terminate) the resource (file or database) where data store permanently.

**finally block important points**

* Finally block will execute compulsory
* Writing finally block is optional.
* You can write finally block for the entire java program
* In some of the circumstances one can also write try and catch block in finally block.

**Example**

**class** ExceptionDemo

{

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

{

**int** a=10, ans=0;

**try**

{

ans=a/0;

}

**catch** (Exception e)

{

System.**out**.println("Denominator not be zero");

}

**finally**

{

System.**out**.println("I am from finally block");

}

}

}

**Output**

Denominator not be zero

I am from finally block