**METHOD**

A method is a set of code which is referred to by name and can be called (invoked) at any point in a program simply by utilizing the method's name.  Think of a method as a subprogram that acts on data and often returns a value.

Each method has its own name.  When that name is encountered in a program, the execution of the program branches to the body of that method.  When the method is finished, execution returns to the area of the program code from which it was called, and the program continues on to the next line of code.

**TYPES OF METHODS IN JAVA**

**Built-in method**: Built in methods are part of the compiler package such as

System.out.println() and system.exit(0)

**User defined method**: user defined methods are created by you the programmer. These method takes on names that you assign to them and perform task you create.

**CATCH AND TRY**

To understand catch and try in java, you need to first understand exception.

**Exception :**The term exception is used to refer to the type of error that one might want to handle with a try..catch. An exception is an exception to the normal flow of control in the program. The term is used in preference to "error" because in some cases, an exception might not be considered to be an error at all. You can sometimes think of an exception as just another way to organize a program.

When an exception occurs, we say that the exception is "thrown". For example, we say that Integer.parseInt (str) throws an exception of type NumberFormatException when the value of str is illegal. When an exception is thrown, it is possible to "catch" the exception and prevent it from crashing the program. This is done with a try..catch statement.

In simplified form, the syntax for a try. Catch can be:

try {

statements-1

}

catch ( exception-class-name variable-name ) {

statements-2

}

**Below is the sample code of how catch and try can be used or implemented in java programs**

package com.javacodegeeks.javabasics.trycatch;

import java.util.ArrayList;

import java.util.List;

public class JavaTryCatchExample {

public static void main(String[] args) {

List<Integer> list = new ArrayList<>();

list.add(3);

list.add(5);

list.add(0);

try{

System.out.println("A list element is: "+list.get(5));

int x = list.get(1);

int y = list.get(2);

System.out.println("Division of x/y is: "+(x/y));

}catch(IndexOutOfBoundsException e){

e.printStackTrace();

}catch(ArithmeticException e){

e.printStackTrace();

}finally{

System.out.println("I will be executed, no matter what :-)");

}

}

}

In the above code, i have commented the statement that caused the exception previously, so as to demonstrate one more type of exception called Arithmetic Exception. In this case, the specific exception occurs because we are trying to divide by zero, which is not allowed. Also, we can observe that there is a finally statement, which will be executed regardless of the occurrence of the exception. If we run the above code; we will have the following results:

Output:

java.lang.ArithmeticException: / by zero at com.javacodegeeks.javabasics.trycatch.JavaTryCatchExample.main(JavaTryCatchExample.java:19)

I will be executed, no matter what :-)

In the results, you can see that the block of code surrounded by the finally statement is executed.