# OBJECT ORIENTED PROGRAMMING USING

Java

Let's explore technology together to live in the future

Checkout more on https://github.com/Sy-hash-collab

Sy-hash-collab

## Java Exception Handling:

### Java Exceptions:

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input. or unforesceable things.
When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an exception (throw an error).

This is why it is important to handle exceptions. Here's a list of approaches:
- try ... catch block.
- finally block
- throw and throws keyword.

### Java try and catch:

The try statement allows you to define a block of code to be tested for errors while it is being executed.
The catch statement allows you to define a block of code to be executed if an error occurs in the try block.
The try and catch keywords come in pairs.

```
try {
  // Block of code to try
}
catch ( Exception e ) {
 // Block of code to handle errors.
}
```

Parent of all java classes.

Built-in , Generic class

Here we have placed the code that might generate an exception inside the try block.
Every try block is followed by a catch block.
When an exception occurs, it is caught by the catch block. The catch block can't be used without try block.
Consider the following example
This will generate an error, because myNumbers [10] does not exist.

```
public class Main {
 public static void main (String [] args)
{  int [] myNumbers = { 1 , 2 , 3 } ;

System.out.println (myNumbers [10]);
  }
}
```

```
Exception in thread
java.lang.ArrayIndexOutOfBoundsException
: 10 .
```

```
public class Main {
public static void main (String[] args)
{ try {
    int[] myNumbers= { 1,2,3 };
  System.out.println (myNumbers [10]);
 }
 catch (Exception e) {
 System.out.println ("Something went wrong");
 }
}
} Something went wrong
```

```
class Main {
public static void main (String[] args)
{ try {
   int divideByZero = 5 / 0;
        }
 catch (ArithmeticException e) {
 System.out.println ("Arithmetic Exception:"
                + e.getMessage ());
}}} Arithmetic Exception / by zero
```

In this example, we're trying to divide a number by 0. Here, this code generates an exception.
To handle exception, we have put the code 5/0 inside try block. Now when an exception occurs, the rest of code inside try block is skipped.
The catch block catches the exception and statements inside catch block is executed.
If none of statements in try block generates an exception, the catch block is skipped.

## 2. Java finally block:

In Java the finally block is always executed no matter whether there's an exception or not.
The finally block is optional. And for each try block, there can be only one finally block.

```
try {
   //code
}
catch (Exception e) {
   //catch block
}
finally {
   //finally block always executes.
}
```

```
public class Main{
 public static void main (String[]args)
{ try {

 int myNumbers = {1,2,3};
 System.out.println (myNumbers [10]);
 }
 catch (Exception e) {

System.out.println ("Something went wrong");
}
 finally {

 System.out.println ("The 'try catch' is
                         finished");
 }
}}
```

```
Something went wrong.
The 'try catch' is finished.
```

```
class Main{
public static void main (String []args)
{
 try {
 int divideByZero =5/0;
 }
catch (ArithmeticException e) {
{ System.out.println ("ArithmeticException
                     + e.getMessage );
 }
  finally{
System.out.println ( "This is finally block");
}}
```

It is a good practice to use the finally block. It's because it can include important cleanup codes like;

- Code that might be accidentally skipped by return , continue or break.
- Closing a file or connection.

In Java, you can create your own custom exception.

```
 try
{
  int [] x = {1,2,4};
  int i = 1;
 if( i>2 && i<=0)
  {throw new ArrayOutOfBoundsException();
  }else
 {
   System.out.println ( x[i]);
 }
}
```

## 3. Java throw and throws keyword:

The Java throw keyword is used explicitly throw a single exception.
The throw statement allows you to create a custom error.

```
public class Main{
 public static void main (String[]args)
{
   try { checkAge(16);
       }
}
 static void checkAge (int age)
{  if (age<18)

  throw new ArithmeticException ("Access
              denied - you must be 18 years
              old");
 else
 System.out.println ("Access granted");
}


public class Main {
 public static void main (String[]args)
{  try {
       checkRollno (44);
       }
}
 static void checkRollno (int rollno)
{ if (rollno<1 && rollno>60 )

 throw new ArithmeticException ("Rollno
          must be b/w 1-60");
```

Example : Exception handling using Java throw.

```
class Main{
public static void main(String args)
{ divideByZero()
 }
 public static void divideByZero()
{ //throw an exception
  throw new ArithmeticException("Tryin
                    to divide by 0");
 }
}
 Exception in thread "main"
  java.lang.ArithmeticException: Trying to
                          divide by o
```

In the above example we're explicitly throwing the ArithmeticException using the throw keyword.

## Note:

When we throw an exception th flow of program moves from the try block to the catch block.

## throws keyword:

the 'throws' keyword used to declare type of exceptions th might occur within the method, used in method declaration.

## Example : Java throws keyword.

```
 import java.io.*;
 class Main {
  public static void main (String[]args)
{  try
  {  findFile();
   }
  catch (IOException e)
  {
    System.out.println(e);
  }
}

 public static void findFile() throws
                    IOException
 {
   File newFile = new File("test.txt"),

 FileInputStream stream = new FileInputStream
                        (newFile);
 }
} java.io.FileNotFoundException : text.txt (The
                   system can't find the file)
```

When we run this program, if file test.txt doesn't exist, the FileInputStream throws a FileNotFoundException which extends the IOException class.

The findFile() method specifies that an IOException can be thrown. The main() method calls this method and handles the exception if it is thrown