Java throw, throws and finally Keyword

throw, throws and finally are the keywords in Java that are used in exception handling. The throw keyword is used to throw an exception and throws is used to declare the list of possible exceptions with the method signature. Whereas finally block is used to execute essential code, specially to release the occupied resources.

1) Java Throw

The throw keyword is used to throw an exception explicitly. Only object of Throwable class or its sub classes can be thrown. Program execution stops on encountering throw statement, and the closest catch statement is checked for matching the type of exception.

Syntax:

throw ThrowableInstance

Creating Instance of Throwable class

We allow to use new operator to create an instance of class Throwable,

new NullPointerException("test");

This constructs an instance of NullPointerException with name test.

Example1 of throw Exception

In this example, we are throwing Arithmetic exception explicitly by using the throw keyword that will be handle by catch block.

```
class Test
{
  static void avg()
  {
    try
```

```
  {
    throw new ArithmeticException("demo");
  }
 catch(ArithmeticException e)
  {
    System.out.println("Exception caught");
  }
}
 public static void main(String args[])
{
  avg();
}
}
```

In the above example the avg() method throw an instance of ArithmeticException, which is successfully handled using the catch statement and thus, the program prints the output "Exception caught".

Example2 of throw Exception

```
class THROW {
   public static void checkNum(int num) {
      if (num < 1) {
        throw new ArithmeticException("Less than One");
      }
      else {
         System.out.println("Square of " + num + " is " + (num*num));
      }
   }
```

```
    public static void main(String[] args) {
        checkNum(-3);
        System.out.println("Exception Handled");
    }
}
```

2) Java throws Keyword

The throws keyword is used to declare the list of exception that a method may throw during execution of program. Any method that is capable of causing exceptions must list all the exceptions possible during its execution, so that anyone calling that method gets a prior knowledge about which exceptions are to be handled. A method can do so by using the throws keyword.

Syntax:

```
type method_name(parameter_list) throws exception_list
{
  // definition of method
}
```

Example1 throws Keyword

Here, we have a method that can throw Arithmetic exception so we mentioned that with the method declaration and catch that using the catch handler in the main method.

```
class Test
{
  static void check() throws ArithmeticException
  {
    System.out.println("Inside check function");
    int a = 20/0;
```

```
   int [] arr = new int[5];
   arr[20]=5;
  }
  public static void main(String args[])
  {
   try
   {
    check();
   }
   catch(ArithmeticException e)
   {
    System.out.println("caught" + e);
   }
  }
}
```

Output

```
Inside check function
caughtjava.lang.ArithmeticException: demo
```

Example2 throws Keyword

```
class Test
{
  static void check() throws
ArithmeticException,ArrayIndexOutOfBoundsException
  {
   System.out.println("Inside check function");
   int a = 20/0;
```

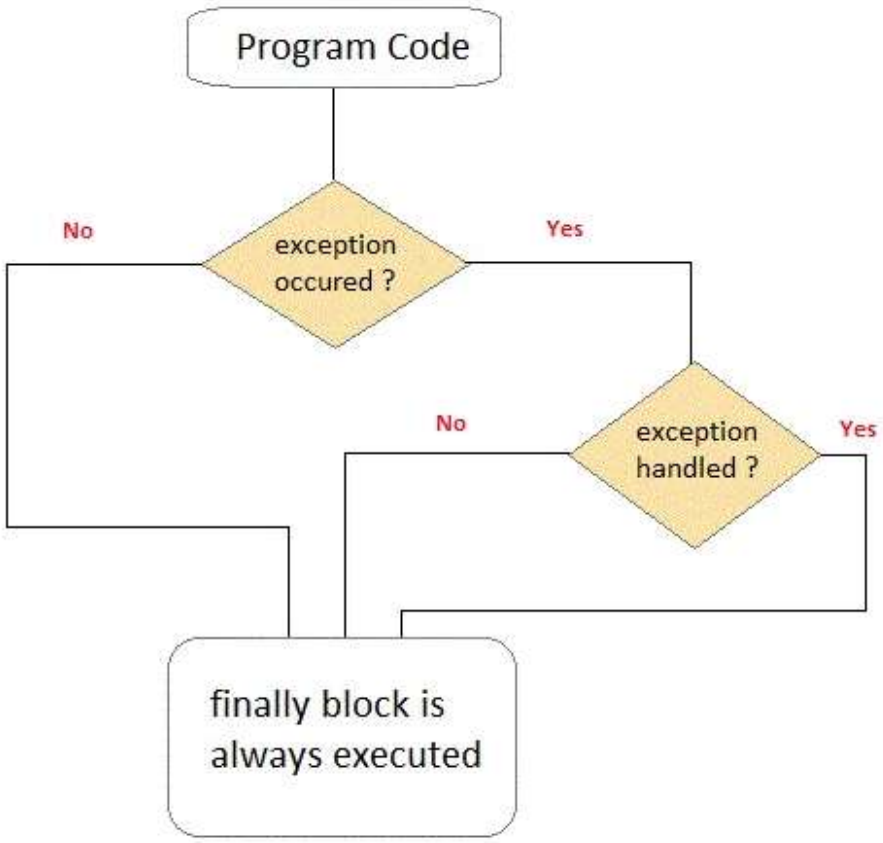
```
  int [] arr = new int[5];
  arr[20]=5;
 }
 public static void main(String args[])
 {
  try
  {
   check();
  }
  catch(ArithmeticException e)
  {
   System.out.println("caught " + "Divide by ZERO");
  }
  catch(ArrayIndexOutOfBoundsException e)
 {
  System.out.println("caught " + "Index out of Bound");
 }
 }
}
```

Difference between throw and throws

throw	throws
throw keyword is used to throw an exception explicitly.	throws keyword is used to declare an exception possible during its execution.
throw keyword is followed by an instance of Throwable class or one of its sub-classes.	throws keyword is followed by one or more Exception class names separated by commas.
throw keyword is declared inside a method body.	throws keyword is used with method signature (method declaration).
We cannot throw multiple exceptions using throw keyword.	We can declare multiple exceptions (separated by commas) using throws keyword.

3) finally clause

A finally keyword is used to create a block of code that follows a try block. A finally block of code is always executed whether an exception has occurred or not. Using a finally block, it lets you run any clean-up type statements that you want to execute, no matter what happens in the protected code. A finally block appears at the end of catch block.



Example1: finally Block

In this example, we are using finally block along with try block. This program throws an exception and due to exception, program terminates its execution but see code written inside the finally block executed. It is because of nature of finally block that guarantees to execute the code.

```
class ExceptionTest {
  public static void main(String[] args)
{
    int a[] = new int[2];
    System.out.println("out of try");
    try
    {
      System.out.println("Access invalid element"+ a[3]);
      /* the above statement will throw ArrayIndexOutOfBoundException */
    }
    finally
    {
      System.out.println("finally is always executed.");
    }
  }
}
```

Output

```
Out of try
finally is always executed.
Exception in thread main java. Lang. exception array Index out of bound exception.
```

You can see in above example even if exception is thrown by the program, which is not handled by catch block, still finally block will get executed.

Example 2: finally Block

finally block executes in all the scenario whether exception is caught or not. In previous example, we use finally where exception was not caught but here exception is caught and finally is used with handler.

```
class Demo
{
  public static void main(String[] args)
  {
    int a[] = new int[2];
    try
    {
System.out.println("Access invalid element"+ a[3]);
      /* the above statement will throw ArrayIndexOutOfBoundException */
    }
    catch(ArrayIndexOutOfBoundsException e) {
      System.out.println("Exception caught");
    }
    finally
    {
      System.out.println("finally is always executed.");
    }
  }
}
```

Output

```
Exception caught
finally is always executed.
```