Throw and Throws

throw Keyword

1.The Java throw keyword is used to throw an exception explicitly.

2.We specify the **exception** object which is to be thrown.

3.The Exception has some message with it that provides the error description.

4. These exceptions may be related to user inputs, server, etc.

5.We can throw either checked or unchecked exceptions in Java by throw keyword.

6.It is mainly used to throw a custom exception.

7.It does not handle the Exception.

8.UserDefineObjects with their own messages can be displayed.

9. Only One Exception can be thrown at a time using keyword throw.

10.throw keyword is mentioned inside the method body.

11. Syntax:

```
 Void method()
{
        Code..
        throw new ExceptionClassName(“user message”);
}
```

```java
public class ThrowKeyword
{
	void method1(int dividend,int divisor)
	{
		if(divisor==0)
		{
	throw new ArithmeticException("Divisor should not be 0");

		}
		else
		{
			int res=dividend/divisor;
			System.out.println("Quotient is:"+res);

		}

	}

	void method2(String s)//s=="abc"
	{
		if(s!="123")
		{
	throw new NumberFormatException("String should be Numeric
value");
		}
		else
		{
			int i=Integer.parseInt(s);

		}


	}


    public static void main(String[] args)
    {
		ThrowKeyword obj=new ThrowKeyword();

		try
		{
			obj.method1(10, 0);
		}
		catch(ArithmeticException e)
		{
			System.out.println(e);
```

```
            }

            try
            {
            obj.method2("abc");
            }
            catch(NumberFormatException e)
            {
                  System.out.println(e);
            }
            System.out.println("Normal Program Execution");

      }

}
```

Output:

java.lang.ArithmeticException: Divisor should not be 0
java.lang.NumberFormatException: String should be Numeric value
Normal Program Execution

throws Keyword

1.The **Java throws keyword** is used to declare an exception.

2. It gives an information to the programmer that there may occur an exception.

3.So, it is better for the programmer to provide the exception handling code so that the normal flow of the program can be maintained.

4.Exception Handling is mainly used to handle the checked exceptions.

5. If there occurs any unchecked exception such as

ArithmeticException, ArrayIndexOutOfBoundsException, NullPointerException,

it is programmers' fault that he is not checking the code before it being used.

6. It can be considered as Signature of the methods.

7. Multiple Exceptions can be Thrown using Keyword “Throws”.

8. Syntax:

void method() throws ExceptionClass1, ExceptionClass2, ExceptionClass3….. ExceptionClass..n

{

Code..

}

```
public class ThrowsKeyword
{
     /*
          void method1() throws ArithmeticException
          {
               int res=5/0;

          }

          void method2() throws NullPointerException
          {
               String k=null;
               System.out.println(k.length());

          }
          //Signature of the Method.
          void method3() throws
ArrayIndexOutOfBoundsException
          {

               int num[]=new int[10];
               /*int[] num2=new int[10];
               int []num3=new int[10];

               num[34]=78;

          }

          void method4() throws
StringIndexOutOfBoundsException
          {
               String s="max";
               System.out.println(s.charAt(20));
          }
```

```
        */


void master() throws ArithmeticException,
NumberFormatException,ArrayIndexOutOfBoundsException,NullP
ointerException
        {
             int val=8/2;

             String s="123";
             int i=Integer.parseInt(s);

             int t[]=new int[10];

             t[2]=100;

             String p="Java";
             System.out.println(p.length());

             String h="Hello";
             System.out.println(h.charAt(70));

        }

    public static void main(String[] args)
    {
         ThrowsKeyword  obj=new ThrowsKeyword();

    /*   try
         {
              obj.method1();//Exception Return
         }
         catch(ArithmeticException e)
         {
              System.out.println(e);
         }

         try
         {
         obj.method2();
         }
         catch(NullPointerException e)
```

```
{
     System.out.println(e);
}

try
{
obj.method3();
}
catch(ArrayIndexOutOfBoundsException e)
{
     System.out.println(e);
}

try
{

     obj.method4();
}
catch(StringIndexOutOfBoundsException e)
{
     System.out.println(e);
}
*/

try
{
     obj.master();
}
catch(ArithmeticException e)
{
     System.out.println(e);
}
catch(NumberFormatException e)
{
     System.out.println(e);
}
catch(ArrayIndexOutOfBoundsException e)
{
     System.out.println(e);
}
catch(NullPointerException e)
{
```

```
                System.out.println(e);
            }
            catch(Exception e)
            {
                System.out.println(e);
            }
            System.out.println("Normal Program Execution");

        }



}
```

Output:

java.lang.StringIndexOutOfBoundsException:
String index out of range: 70
Normal Program Execution

Difference Between Throw and Throws

S. No.	Key Difference	throw	throws
1.	Point of Usage	The **throw** keyword is used inside a function. It is used when it is required to throw an Exception logically.	The **throws** keyword is used in the function signature. It is used when the function has some statements that can lead to exceptions.
2.	Exceptions Thrown	The **throw** keyword is used to throw an exception explicitly. It can throw only one exception at a time.	The **throws** keyword can be used to declare multiple exceptions, separated by a comma. Whichever exception occurs, if matched with the declared ones, is thrown automatically then.
3.	Syntax	Syntax of **throw** keyword includes the instance of the Exception to be thrown. Syntax wise throw keyword is followed by the instance variable.	Syntax of **throws** keyword includes the class names of the Exceptions to be thrown. Syntax wise throws keyword is followed by exception class names.
4.	Propagation of Exceptions	**throw** keyword cannot propagate checked exceptions. It is only used to propagate the unchecked Exceptions that are not checked using the throws keyword.	**throws** keyword is used to propagate the checked Exceptions only.