throw and throws in Java

Difficulty Level : Easy Last Updated : 26 Jan, 2021

throw

The throw keyword in Java is used to explicitly throw an exception from a method or any block of code. We can throw either checked or unchecked exception. The throw keyword is mainly used to throw custom exceptions.

Syntax:

```
throw Instance
Example:
throw new ArithmeticException("/ by zero");
```

But this exception i.e, *Instance* must be of type **Throwable** or a subclass of **Throwable**. For example Exception is a sub-class of Throwable and user defined exceptions typically extend Exception class. Unlike C++, data types such as int, char, floats or non-throwable classes cannot be used as exceptions.

The flow of execution of the program stops immediately after the throw statement is executed and the nearest enclosing **try** block is checked to see if it has a **catch** statement that matches the type of exception. If it finds a match, controlled is transferred to that statement otherwise next enclosing **try** block is checked and so on. If no matching **catch** is found then the default exception handler will halt the program.

```
// Java program that demonstrates the use of throw
class ThrowExcep
{
    static void fun()
    {
        try
        {
            throw new NullPointerException("demo");
        }
        catch(NullPointerException e)
        {
            System.out.println("Caught inside fun().");
```

```
            throw e; // rethrowing the exception
        }
    }

    public static void main(String args[])
    {
        try
        {
            fun();
        }
        catch(NullPointerException e)
        {
            System.out.println("Caught in main.");
        }
    }
}
```

Output:

```
Caught inside fun().
Caught in main.
```

Another Example:

```
// Java program that demonstrates the use of throw
class Test
{
    public static void main(String[] args)
    {
        System.out.println(1/0);
    }
}
```

Output:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

throws

throws is a keyword in Java which is used in the signature of method to indicate that this method might throw one of the listed type exceptions. The caller to these methods has to handle the exception using a try-catch block.

Syntax:

```
type method_name(parameters) throws exception_list
exception_list is a comma separated list of all the
exceptions which a method might throw.
```

In a program, if there is a chance of raising an exception then compiler always warn us about it and compulsorily we should handle that checked exception, Otherwise we will get compile time error saying **unreported exception XXX must be caught or declared to be thrown**. To prevent this compile time error we can handle the exception in two ways:

1. By using try catch
2. By using **throws** keyword

We can use throws keyword to delegate the responsibility of exception handling to the caller (It may be a method or JVM) then caller method is responsible to handle that exception.

```
// Java program to illustrate error in case
// of unhandled exception
class tst
{
    public static void main(String[] args)
    {
        Thread.sleep(10000);
        System.out.println("Hello Geeks");
    }
}
```

Output:

```
error: unreported exception InterruptedException; must be caught or declared
```

Explanation: In the above program, we are getting compile time error because there is a chance of exception if the main thread is going to sleep, other threads get the chance to execute main() method which will cause InterruptedException.

```
// Java program to illustrate throws
class tst
{
    public static void main(String[] args)throws InterruptedException
    {
        Thread.sleep(10000);
        System.out.println("Hello Geeks");
    }
}
```

Output:

```
Hello Geeks
```

Explanation: In the above program, by using throws keyword we handled the InterruptedException and we will get the output as **Hello Geeks**

Another Example:

```
// Java program to demonstrate working of throws
class ThrowsExecp
{
    static void fun() throws IllegalAccessException
    {
        System.out.println("Inside fun(). ");
        throw new IllegalAccessException("demo");
    }
    public static void main(String args[])
    {
        try
        {
            fun();
        }
```

```
            catch(IllegalAccessException e)
            {
                System.out.println("caught in main.");
            }
        }
    }
```

Output:

```
Inside fun().
caught in main.
```

Important points to remember about throws keyword:

- throws keyword is required only for checked exception and usage of throws keyword for unchecked exception is meaningless.
- throws keyword is required only to convince compiler and usage of throws keyword does not prevent abnormal termination of program.
- By the help of throws keyword we can provide information to the caller of the method about the exception.

Reference: Java – The complete Reference by Herbert Schildt

This article is contributed by **Pratik Agarwal and Bishal Dubey**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.