

Java - Throws and Throw | Throw an Exception

Java throws and throw

If a **method** does not handle a checked **exception**, the method must declare it using the **throws** keyword. The throws keyword appears at the end of a method's signature.

You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the **throw** keyword.

Try to understand the difference between throws and throw keywords, throws is used to postpone the handling of a checked exception and throw is used to invoke an exception explicitly.

Syntax

Following is the syntax of throwing an exception using **throws** and **throw** -

```
method(parameters) throws exception {
    // Method implementation
    throw new exception();
}
```

The following method declares that it throws a RemoteException –

Consider the below example code to use **throws** and **throw** keywords -

```
import java.io.*;
public class className {

    public void deposit(double amount) throws RemoteException {
        // Method implementation
        throw new RemoteException();
    }
    // Remainder of class definition
}
```

A method can declare that it throws more than one exception, in which case the exceptions are declared in a list separated by commas. For example, the following

method declares that it throws a `RemoteException` and an `InsufficientFundsException` –

```
import java.io.*;
public class className {

    public void withdraw(double amount) throws RemoteException,
        InsufficientFundsException {
        // Method implementation
    }
    // Remainder of class definition
}
```

Learn **Java** in-depth with real-world projects through our **Java certification course**. Enroll and become a certified expert to boost your career.

Java Throws and Throw Example

Following example shows the use of `throw` keyword to send an exception in case a invalid argument is passed. We're calling a `divide` method which checks if second parameter is zero then it will throw an `IllegalArgumentException` with a custom message. As `IllegalArgumentException` is an unchecked exception, the `divide` method is not required to declares `throws` statement. Now as parent method is not handling the exception, JVM intercepts the same and prints the error message and terminates the program.


[Open Compiler](#)

```
package com.tutorialspoint;

public class ExcepTest {

    public static void main(String args[]) {
        int a = 3;
        int b = 0;
        System.out.println("result:" + divide(a,b));
    }

    private static int divide(int a, int b) {
        if(b == 0) {
            throw new IllegalArgumentException("second argument cannot be
zero.");
        }
        return a / b;
    }
}
```

```
}  
}
```

Output

```
Exception in thread "main" java.lang.IllegalArgumentException: second argument cannot be zero  
    at com.tutorialspoint.ExcepTest.divide(ExcepTest.java:13)  
    at com.tutorialspoint.ExcepTest.main(ExcepTest.java:8)
```

More Examples

Example 1: Throw an exception on invalid arguments

Following example shows the use of throw and throws keywords to send an exception in case a invalid argument is passed and handle the exception. We're calling a divide method which checks if second parameter is zero then it will throw an Exception with a custom message. As Exception is a checked exception, the divide method is required to declares throws statement. Now as parent method is to either handle the exception or declares the throws exception, we're handling the exception and printing the message.


[Open Compiler](#)

```
package com.tutorialspoint;  
  
public class ExcepTest {  
  
    public static void main(String args[]) {  
        int a = 3;  
        int b = 0;  
        try {  
            System.out.println("result:" + divide(a,b));  
        } catch (Exception e) {  
            System.out.println("Exception: " + e);  
        }  
    }  
  
    private static int divide(int a, int b) throws Exception {  
        if(b == 0) {  
            throw new Exception("second argument cannot be zero.");  
        }  
    }  
}
```

```
    }  
    return a / b;  
  }  
}
```

Output

Exception: java.lang.Exception: second argument cannot be zero.

Example 2: Using throws and throw in main and other method

Following example shows the use of throw and throws keywords to send an exception in case a invalid argument is passed and exception is not handled. We're calling a divide method which checks if second parameter is zero then it will throw an Exception with a custom message. As Exception is a checked exception, the divide method is required to declares throws statement. Now as parent method is to either handle the exception or declares the throws exception, we're declaring to throw the exception and JVM will handle the exception.

[Open Compiler](#)

```
package com.tutorialspoint;  
  
public class ExcepTest {  
  
    public static void main(String args[]) throws Exception {  
        int a = 3;  
        int b = 0;  
        System.out.println("result:" + divide(a,b));  
    }  
  
    private static int divide(int a, int b) throws Exception {  
        if(b == 0) {  
            throw new Exception("second argument cannot be zero.");  
        }  
        return a / b;  
    }  
}
```

Output

```
Exception in thread "main" java.lang.Exception: second argument cannot be zero.  
    at com.tutorialspoint.ExcepTest.divide(ExcepTest.java:15)  
    at com.tutorialspoint.ExcepTest.main(ExcepTest.java:9)
```