In the previous article [**Java try/catch Block**](http://www.javaguides.net/2018/08/java-trycatch-block.html), we have only been catching exceptions that are thrown by the Java run-time system. However, it is possible for your program to throw an exception explicitly, using the *throw*statement. The general form of throw:

throw ThrowableInstance;

Here, *ThrowableInstance* must be an object of type **[Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html" \t "_blank)** or a subclass of **[Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html" \t "_blank)**.

Primitive types, such as *int* or *char*, as well as non-Throwable classes, such as [**String**](http://www.javaguides.net/p/java-string-api-guide.html) and *Object*, cannot be used as exceptions.

There are two ways you can obtain a **[Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html" \t "_blank)** object:

* using a parameter in a *catch*clause.
* creating one with the *new*operator.

The flow of execution stops immediately after the *throw*statement; any subsequent statements are not executed. The nearest enclosing [**try**](http://www.javaguides.net/2018/08/java-trycatch-block.html) block is inspected to see if it has a *catch*statement that matches the type of exception. If it does find a match, control is transferred to that statement. If not, then the next enclosing try statement is inspected, and so on. If no matching *catch*is found, then the default exception handler halts the program and prints the stack trace.

Here is a sample program that creates and *throws*an exception. The handler that catches the exception rethrows it to the outer handler.

**Java throw Keyword Examples**

**Using throw Keyword Example 1**

Let's create custom exception *ResourceNotFoundException* and use *throw* keyword is used to explicitly throw an exception.

package com.javaguides.exceptions.examples;

public class TestResourceNotFoundException {

public static void main(String[] args) throws ResourceNotFoundException {

ResourceManager manager = new ResourceManager();

manager.getResource(0);

}

}

class Resource {

private int id;

public Resource(int id) {

super();

this.id = id;

}

}

class ResourceManager {

public Resource getResource(int id) throws ResourceNotFoundException {

if (id == 10) {

new Resource(id);

} else {

throw new ResourceNotFoundException("Resource not found with id ::" + id);

}

return null;

}

}

class ResourceNotFoundException extends Exception {

private static final long serialVersionUID = 1L;

public ResourceNotFoundException(Object resourId) {

super(resourId != null ? resourId.toString() : null);

}

}

Output:

Exception in thread "main" com.javaguides.exceptions.examples.ResourceNotFoundException: Resource not found with id ::0

at com.javaguides.exceptions.examples.ResourceManager.getResource(TestResourceNotFoundException.java:26)

at com.javaguides.exceptions.examples.TestResourceNotFoundException.main(TestResourceNotFoundException.java:6)

Note that we have used below code to demonstrate usage of *throw* keyword.

throw new ResourceNotFoundException("Resource not found with id ::" + id);

**Using throw Keyword Example 2**

Here is a sample program that creates and throws an exception. The handler that catches the exception rethrows it to the outer handler.

package com.javaguides.exceptions.examples;

//Demonstrate throw.

class ThrowDemo {

static void demoproc() {

try {

throw new NullPointerException("demo");

} catch (NullPointerException e) {

System.out.println("Caught inside demoproc.");

throw e; // rethrow the exception

}

}

public static void main(String args[]) {

try {

demoproc();

} catch (NullPointerException e) {

System.out.println("Recaught: " + e);

}

}

}

This program gets two chances to deal with the same error. First, *main( )* sets up an exception context and then calls *demoproc( )*. The *demoproc( )* method then sets up another exception-handling context and immediately throws a new instance of *NullPointerException*, which is caught on the next line. The exception is then rethrown. output:

Caught inside demoproc.

Recaught: java.lang.NullPointerException: demo