Rex Jones II

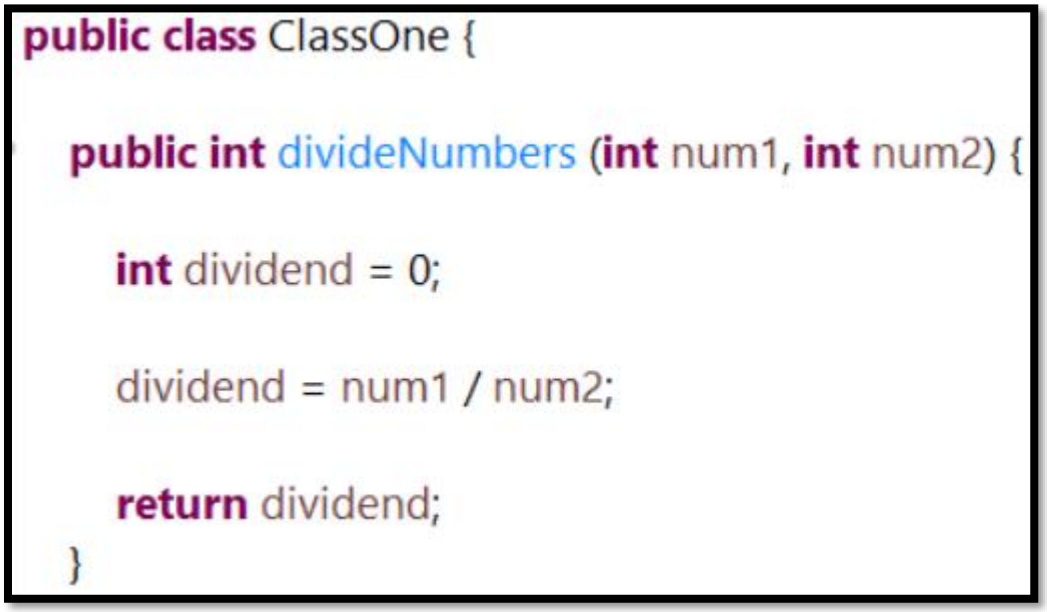
(Transcript)
Keyword Throws

Rex Jones II

Keyword Throws In Java

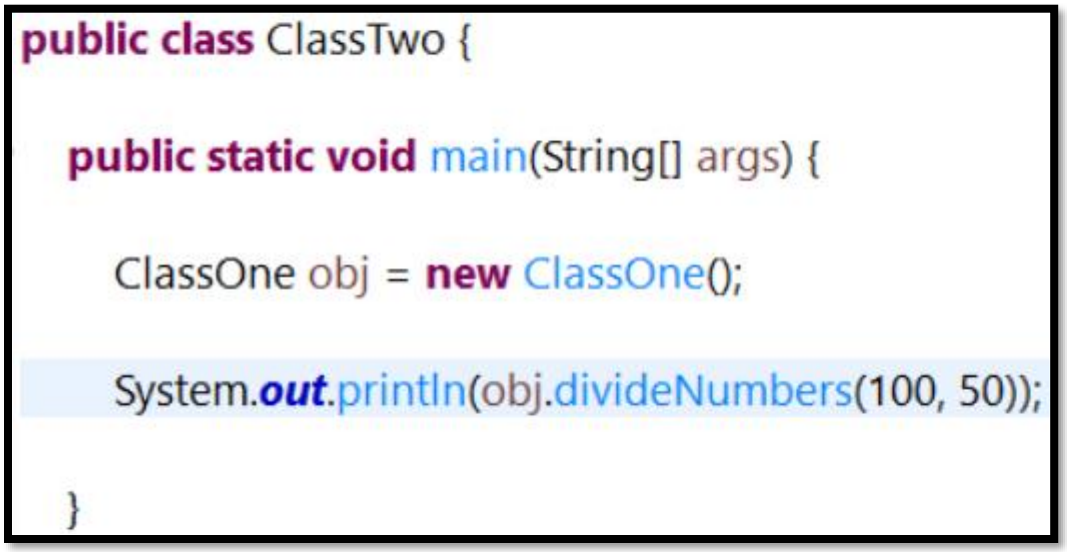
When it comes to the keyword throws, we must declare 1 or more exceptions in a throws clause if the method generates an exception and does not handle the exception. The responsibility of handling the exception falls onto the calling method. In the previous video, I mentioned the keyword throw is used to manually throw an exception.

Let's look at throw and throws. Under ClassOne, I have a method called divideNumbers that receive 2 integer arguments. num1 gets divided by num2 and the value is assigned to dividend. Dividend is returned to the calling method in ClassTwo.



```
public class ClassOne {

   public int divideNumbers (int num1, int num2) {

      int dividend = 0;

      dividend = num1 / num2;

      return dividend;
   }
```

We see under ClassTwo, I created an object called obj. obj calls the divideNumbers method in ClassOne then pass 100 and 50.



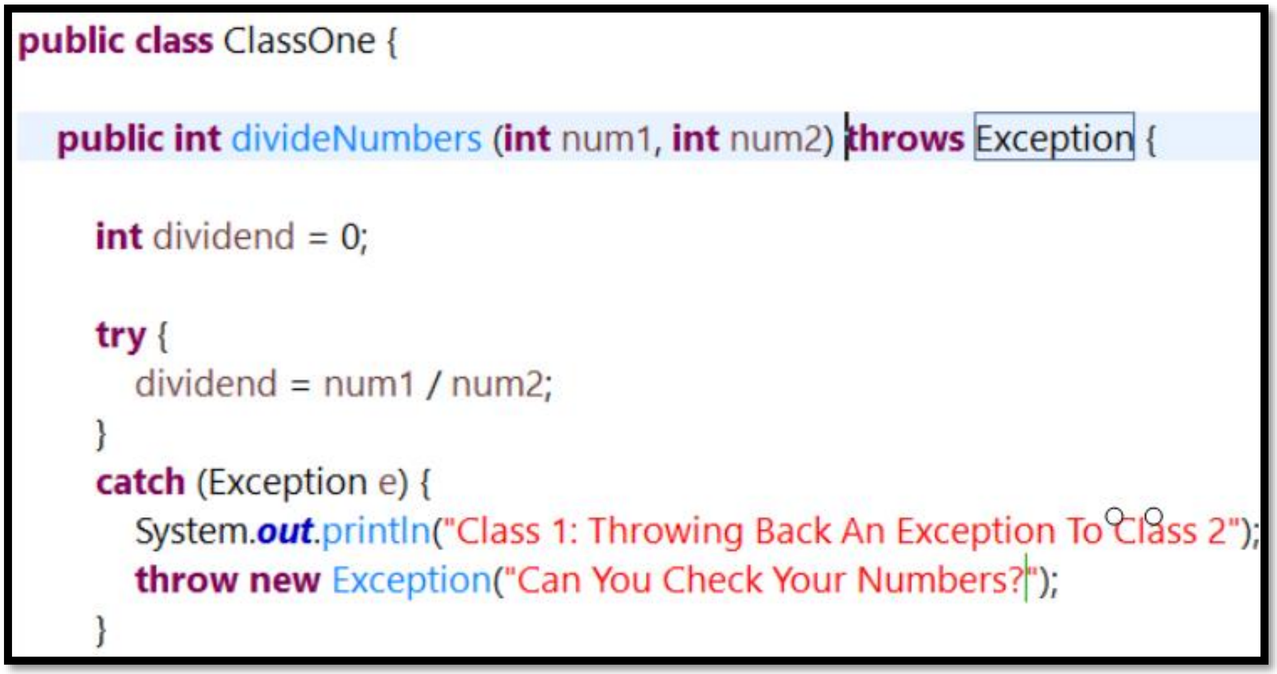
```
public class ClassTwo {

   public static void main(String[] args) {

      ClassOne obj = new ClassOne();

      System.out.println(obj.divideNumbers(100, 50));

   }
```

When I execute, we see 2 is returned to the console.

Suppose, I change 50 to 0. In the previous video, we saw an exception will show up because a number cannot be divided by 0. Let's Run again. Did the exception happen in ClassOne or ClassTwo? The stack trace shows us ArithmeticException divided by zero and ClassOne.divideNumbers. When this happened, code was running in ClassOne. However, ClassTwo must handle the exception. ClassTwo must handle the exception because it sent the values: 100 and 0.

Let’s handle the exception by surrounding dividend = num1 / num2 with a try statement. In the catch statement (Exception e), we write a print statement sysout(“Class 1: Throwing Back An Exception To Class 2”). At this point, ClassOne does not know how to handle the data because ClassTwo is sending data that’s causing a problem. Let’s rethrow the exception by throwing back an exception to the calling class with a customized message: throw new Exception(“Can You Check Your Numbers?”).

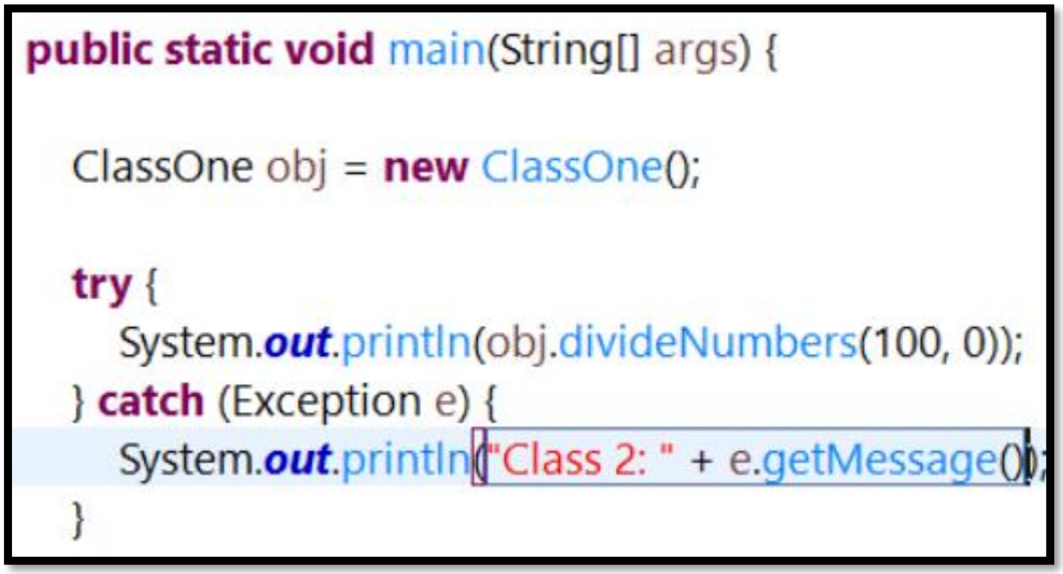
Here’s where we implement the throws keyword. Notice, there’s a compile error. Hover the error, and we see add throws declaration. Select it and throws Exception is added to our method signature.



```
public class ClassOne {

  public int divideNumbers (int num1, int num2) throws Exception {

    int dividend = 0;

    try {
      dividend = num1 / num2;
    }
    catch (Exception e) {
      System.out.println("Class 1: Throwing Back An Exception To Class 2");
      throw new Exception("Can You Check Your Numbers?");
    }
```

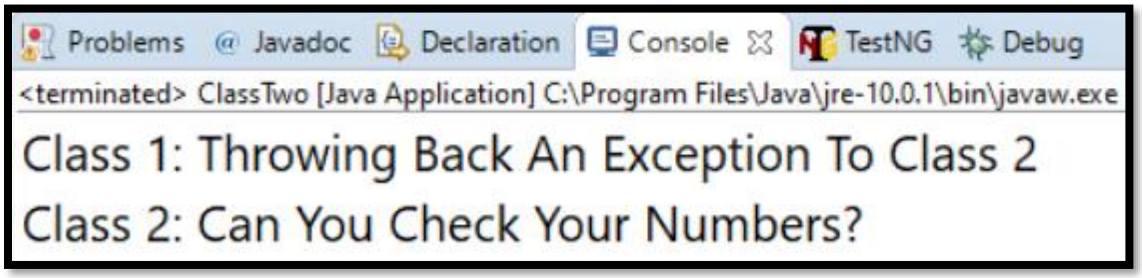
Saving ClassOne causes an error in ClassTwo. ClassTwo has an error because it’s calling the divideNumbers method. The divideNumbers method indicates it may throw an exception. Therefore, ClassTwo has to handle the exception that ClassOne throws back. Hover the error and select surround with try/catch. Print the message: sysout(“Class 2: ” + e.getMessage).

Rex Jones II



```
public static void main(String[] args) {

    ClassOne obj = new ClassOne();

    try {
        System.out.println(obj.divideNumbers(100, 0));
    } catch (Exception e) {
        System.out.println("Class 2: " + e.getMessage());
    }
```

Now, let's run. We see Class 1: Throwing Back An Exception / Class 2: Can You Check Your Numbers?



```
<terminated> ClassTwo [Java Application] C:\Program Files\Java\jre-10.0.1\bin\javaw.exe
Class 1: Throwing Back An Exception To Class 2
Class 2: Can You Check Your Numbers?
```

To recap, Class 2 calls the divideNumbers method in Class 1. Class 1 receive values (100 and 0) which causes an exception. It tries the numbers then catch the exception. After catching the exception, Class 1 throws a new exception with a message back to Class 2. The throws keyword is used in the method's signature because the method may throw an Exception. We can add 1 or more exceptions separated by a comma. That's it for throw and throws.

Social Media Contact

- YouTube https://www.youtube.com/c/RexJonesII/videos
- Twitter https://twitter.com/RexJonesII
- LinkedIn https://www.linkedin.com/in/rexjones34/
- GitHub https://github.com/RexJonesII/Free-Videos
- Facebook http://facebook.com/JonesRexII