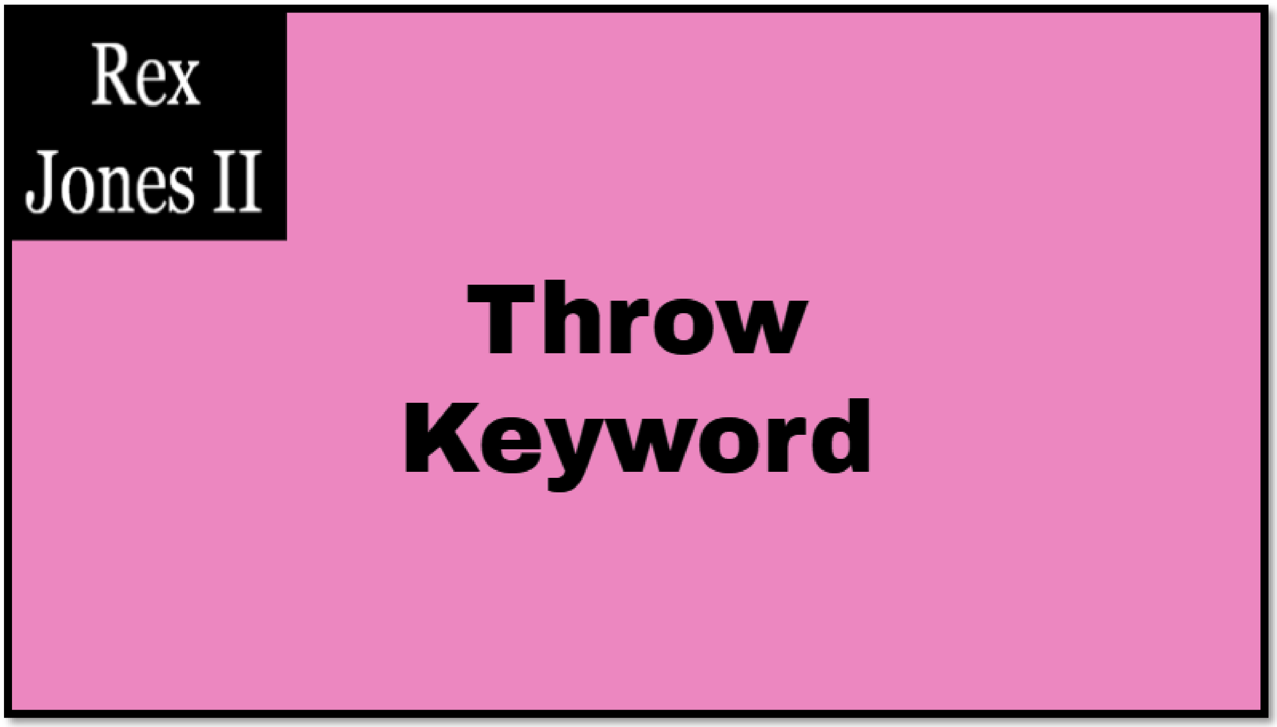
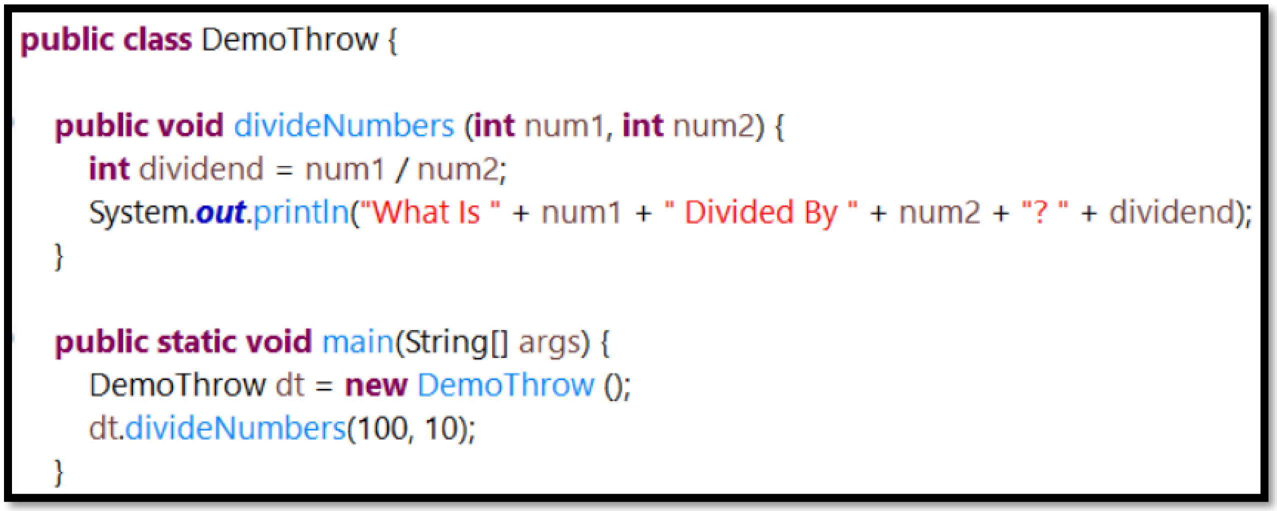
(Transcript)
Keyword Throw



Link To Watch Video = https://youtu.be/OBYFRNj2o04

The keyword throw is used to manually throw an exception. Videos 81 and 82, Introduce you to Exceptions and How To Handle the Exceptions. An exception shows there is a problem with our program. So, why do we want to manually throw an exception if it shows there is a problem? Manually throwing an exception allows us to handle the problem in our code as part of the overall handling exception strategy. Most of the times, the exceptions we throw, will be an instance of an exception class that we create.

In Eclipse, I have a class called DemoThrow with 2 methods: divideNumbers and the main method. The divideNumbers method has int dividend = num1 / num2 and a print statement. The main method will pass 100 to num1 and 10 to num2.

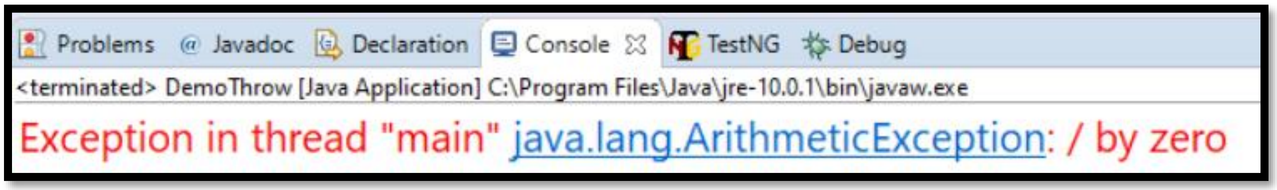


```
public class DemoThrow {

  public void divideNumbers (int num1, int num2) {
    int dividend = num1 / num2;
    System.out.println("What Is " + num1 + " Divided By " + num2 + "? " + dividend);
  }

  public static void main(String[] args) {
    DemoThrow dt = new DemoThrow ();
    dt.divideNumbers(100, 10);
  }
```

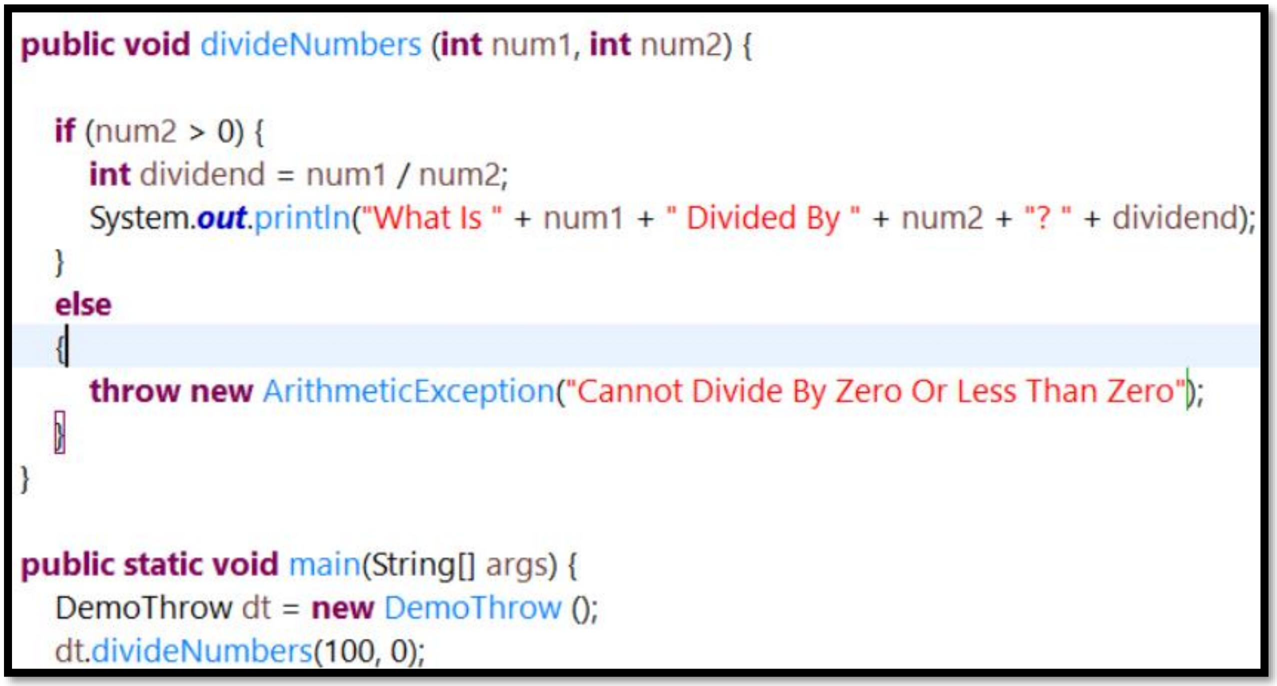
Let's Run. The Console shows What Is 100 Divided By 10? 10 is the answer. That's cool. What if we pass 0 as num2 then Run? Now, we see as ArithmeticException / zero.



```
<terminated> DemoThrow [Java Application] C:\Program Files\Java\jre-10.0.1\bin\javaw.exe
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

Manually throwing an exception or we can say explicitly throwing an exception allows us to handle this problem and customize the exception. Place these 2 statements in an if block. Write

```
if (num2 > 0) {
  int dividend = num1 / num2;
  sysout("What Is " + num1 + " Divided By " + num2 + "? " + dividend)
}
else {
  throw new ArithmeticException("Cannot Divide By Zero Or Less Than Zero")
}
```

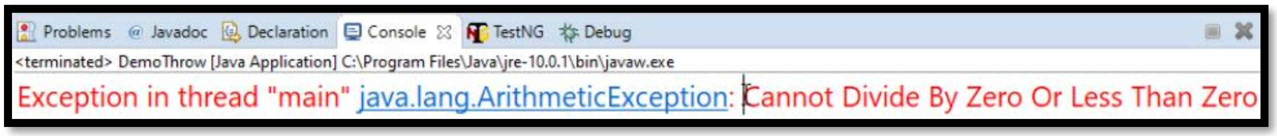


```
public void divideNumbers (int num1, int num2) {

  if (num2 > 0) {
    int dividend = num1 / num2;
    System.out.println("What Is " + num1 + " Divided By " + num2 + "? " + dividend);
  }
  else
  {
    throw new ArithmeticException("Cannot Divide By Zero Or Less Than Zero");
  }
}

public static void main(String[] args) {
  DemoThrow dt = new DemoThrow ();
  dt.divideNumbers(100, 0);
```

First, we write keyword throw then new followed by the exception ArithmeticException. Customize the exception by writing “Cannot Divide By Zero Or Less Than Zero”. Run again. We see the same ArithmeticException with a customize message “Cannot Divide By Zero Or Less Than Zero”.



```
<terminated> DemoThrow [Java Application] C:\Program Files\Java\jre-10.0.1\bin\javaw.exe
Exception in thread "main" java.lang.ArithmeticException: Cannot Divide By Zero Or Less Than Zero
```

This was an intro to keyword throw in Java. I plan to show you how 2 keywords throw and throws work together to make our program complete.

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