Due:
Activity (in-lab): Monday, November 12, 2012 by the end of lab

Goals:

By the end of this project you should be able to do the following:

- Handle exceptions that can occur in your program using a try-catch statement
- Throw exceptions in your program

Directions:

For this assignment, you will be creating two classes: Division and DivisionDriver

Division: methods (15%)

- Division has two public, static methods:
  - intDivide: takes two **int** parameters (a numerator and denominator), performs integer division, and returns the **int** result of dividing the numerator by the denominator.
  - decimalDivide: takes two **int** parameters (a numerator and denominator), performs floating point division (you'll have to use casting), and returns the result of dividing the numerator by the denominator.

- Test your methods in the interactions pane:

```
Division.intDivide(10, 3)
3
Division.decimalDivide(10, 3)
3.3333333333333335
```

DivisionDriver (35%)

- DivisionDriver contains a main method only. The program will get a numerator and denominator from the user and print the integer division and decimal division result.

- Create a **dialog box** that will get the numerator and denominator as a String (you'll have to import the JOptionPane class in the javax.swing package):

```
String numInput
   = JOptionPane.showInputDialog("Enter the numerator:");

String denomInput
   = JOptionPane.showInputDialog("Enter the denominator:");
```

- Convert each to an integer value using the static parseInt method in the Integer class:

```
int num = _________._________(numInput);
int denom = _________._________ (denomInput);
```

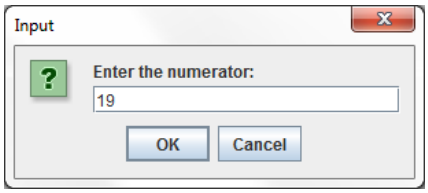
- Create a String object to hold the result of the division:

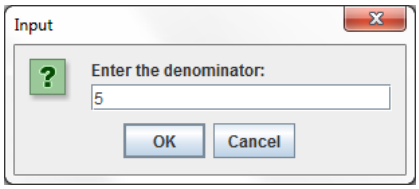
```
String result = "Integer division: \r\n"
   + Division.intDivide(num, _________)
   + "\r\n\r\nFloating point division: \r\n"
   + Division._________(num, denom);
```

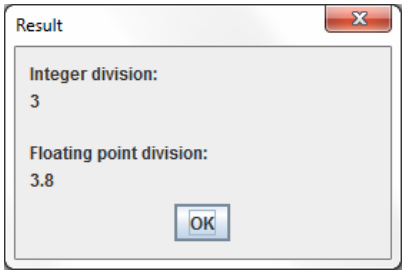
- Print the result in a dialog box:

```
JOptionPane.showMessageDialog(null, result,
   "Result", JOptionPane.PLAIN_MESSAGE);
```

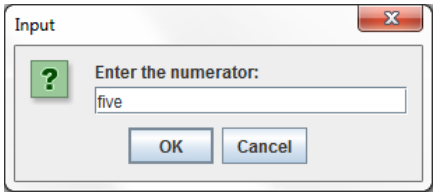
- Test your method by running the driver program with numerator 19 and denominator 5:

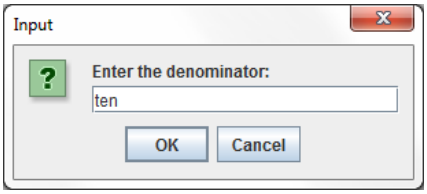






- Now try entering an invalid number in the dialogs (five and ten):





Your program should generate a run-time error in the form of a NumberFormatException exception:

```
----jGRASP exec: java DivisionDriver

Exception in thread "main" java.lang.NumberFormatException: For input string: "five"
        at java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)
        at java.lang.Integer.parseInt(Integer.java:449)
        at java.lang.Integer.parseInt(Integer.java:499)
        at DivisionDriver.main(DivisionDriver.java:12)
```

```
----jGRASP wedge2: exit code for process is 1.
----jGRASP: operation complete.
```

- The exception occurs when the parseInt method tries to convert the String "five" to an integer. The Java API listing for parseInt lists the exception that it might throw.

- Use a try-catch statement to catch the exception and tell the user what went wrong without creating a run-time error:

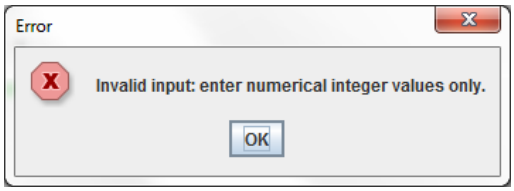
```
try {
 int num = Integer.parseInt(numInput);
 int denom = Integer.parseInt(denomInput);

 String result = "Integer division: \r\n"
    + Division.intDivide(num, denom)
    + "\r\n\r\nFloating point division: \r\n"
    + Division.decimalDivide(num, denom);

  JOptionPane.showMessageDialog(null, result,
    "Result", JOptionPane.PLAIN_MESSAGE);
}
 catch (NumberFormatException errorDetail) {
     JOptionPane.showMessageDialog(null,
     "Invalid input: enter numerical integer values only.",
      "Error", JOptionPane.ERROR_MESSAGE);
 }
```

Existing code

- Try entering invalid values five and ten once more for numerator and denominator once more. You should now get the following error:

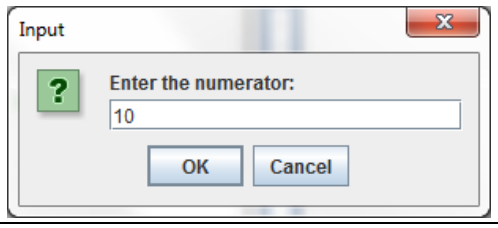


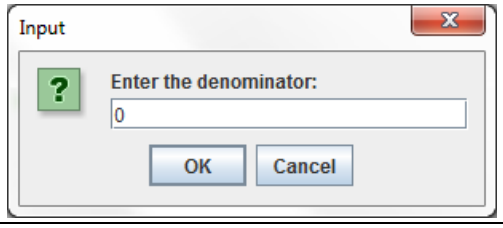
Exception Throwing (35%)

- Try the following in the interactions pane:

```
Division.intDivide(10, 0)
Exception in evaluation thread java.lang.ArithmeticException: / by zero
```

- Try to run your driver program with the numerator 10 and denominator 0:






```
----jGRASP exec: java DivisionDriver

Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Division.intDivide(Division.java:4)
        at DivisionDriver.main(DivisionDriver.java:17)

----jGRASP wedge2: exit code for process is 1.
----jGRASP: operation complete.
```

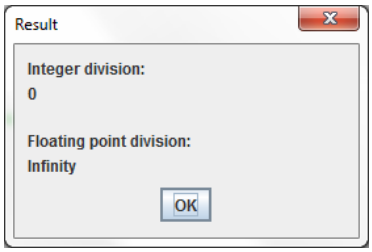
- The exception is generated in the intDivide method and thrown to main via **exception propagation**. This time the error before it is propagated rather than in the main method.

- In your intDivide method, add code that will return 0 if an ArithmeticException occurs and the division result otherwise:

```
try {
    return num / denom;
}
    catch (_________________ error) {
      return _________;
    }
```

Existing code

- Run DivisionDriver with inputs 10 and 0. The result should be 0 for integer division:



- Suppose that you do not want users to be able to divide by 0 in your decimalDivide method.

```
Division.decimalDivide(10, 0)
Infinity
```

The IllegalArgumentException in the Java API can be thrown if a particular argument (parameter) to a method is not allowed:
http://download.oracle.com/javase/6/docs/api/java/lang/IllegalArgumentException.html

- In your decimalDivide method, throw an IllegalArgumentException if the denominator is zero:

```
if (denom == 0) {
    throw new IllegalArgumentException("The denominator "
        + "cannot be zero.");
}
```

- Test your method again in interactions. You should now see the exception:

```
Division.decimalDivide(10, 0)
Exception in evaluation thread
java.lang.IllegalArgumentException: The denominator cannot
be zero.
```

- In your main method, add another catch statement to catch the exception that is thrown by the decimalDivide method. This time, print the exception text itself (stored by variable errorMessage:
-

```
    catch (NumberFormatException errorDetail) {
      JOptionPane.showMessageDialog(null,
       "Invalid input: enter numerical integer values only.",
       "Error", JOptionPane.ERROR_MESSAGE);
    }
    catch (_________________ errorMessage) {
      JOptionPane.showMessageDialog(null,
        errorMessage, "Error", JOptionPane.ERROR_MESSAGE);
    }
```

Existing code

- Now try dividing by 0 in your program. You will get the following error message instead of a run-time error: