In this example, a switch statement was used in the catch block to look at all possible exception messages. A switch statement accomplishes the same task as an embedded if statement. You could have used:

```
If($e->getMessage == "Zero Exception")
{ echo "The value of second number must be greater than zero"; }
else if($e->getMessage == "Some other exception")
{ echo "You did something else wrong"; }
else
{ echo $e->getMessage(); }
```

For some, the switch statement is easier to understand when looking at multiple possible values for the same property (variable) or the result of executing a method (as in this example). The default section of the switch statement (or the last else statement in the embedded if statement) catches anything you did not anticipate. In this example, you simply display the exception message for other exceptions.

As stated earlier, it's very important that you handle all exceptions and errors. By including the default code you are able to handle exceptions and errors you may have never anticipated. Notice that each case section must include a break as the last statement. This keeps the code from following through into the next case statement.

```
Catch(Exception $e) {
        switch($e->getMessage()) {
                case "Zero Exception":
                        echo "The value of second number must be greater than zero";
                case "Some other exception":
                        echo "You did something else wrong";
                        break;
                default:
                        echo $e->getMessage():
        }
```

In this example, if the Zero Exception occurred, both of the messages ("The value of the second number must be greater than zero" and "You did something else wrong") would be displayed. The use of the switch statement is very common in catch blocks. However, as stated earlier, you can use the embedded if statement if you prefer.

Another way you can handle multiple exceptions is to create your own exceptions, throw them, and then catch them. You will need to create a class for your own exception.

```
class zeroException extends Exception {
   public function errorMessage() {
         $errorMessage = "Second Number cannot be " . $this->getMessage();
         return $errorMessage;
         }
 }
try {
if ($secondNumber == 0)
{ throw new zeroException("Zero"); }
else
```