Java Bytes 2

Review: A **block** of code is a group of statements enclosed in curly braces { }.

Throwing (and Catching) Exceptions

An exception is a run-time error. Some problem occurs when the program is running. Examples of common problems are: division by zero, invalid input, array index is out of bounds, attempt to use a null reference, etc. I'm sure that you've come across a variety of exceptions in your Java programming experience.

If we just write code without considering possible exceptions, when we run the program, the program breaks (stops abruptly). That's not good. (Sports analogy: Can you imagine what would happen if a basketball game was over...no winner, no loser...just over... the first time a player committed a foul?)

Exception handling is a feature of many object-oriented programming languages. It's a way to plan for, and deal with, possible problems that may occur when the program is running. By using exception handling, we can keep the program running, maybe not for a long time, but at least long enough to print a helpful message to the user.

There are four Java reserved words involved in working with exceptions.

```
throw
try
catch
finally
```

The **throw** statement allows you to throw an exception and provide an error message. This causes an exception to occur. Remember: If an exception occurs and we don't have any statements in our program to deal with it, the program is over.

The **try** statement allows you to write a block of code that potentially could cause an exception. The try statement is a way of watching to see what happens with those lines of code. This piece of code is often called a "try block".

The **catch** statement allows you to write a block of code that will execute if an exception does occur in the try block. If no exception occurs, the catch statement is ignored and the program goes on. There may be more than one catch statement for a try block.

The **finally** statement allows you to write a block of code that will be executed no matter what happens in the try block. Note: The finally statement will not be used in Lab2. I just wanted you to know what it's used for.

More explanation can be found here: https://www.w3schools.com/java/java_try_catch.asp

The Ternary Conditional

The ternary (ternary means 3) conditional is a quick way to embed a condition (rather like a mini if-statement) inside another statement. Remember that an if statement requires a boolean condition and the values of this condition can only be true or false.

A ternary conditional expression has 3 operands and evaluates to a single value. Let's look at the syntax:

```
condition ? exprTrue : exprFalse
```

When the condition is true, the expression evaluates to `exprTrue`. When the condition is false, the expression evaluates to `exprFalse`. A ternary conditional can be placed inside another statement, anywhere that a value of the same type as exprTrue and exprFalse could be placed.

Example:

```
Scanner scan = new Scanner(System.in);
System.out.println("Enter an integer and I will tell you whether it is odd or
even.");
int num = scan.nextInt();
System.out.println( num % 2 == 0 ? "even" : "odd");
```

The ternary conditional in the print statement is a shorter, but less readable, way to write this if-else statement:

```
if (num % 2 == 0)
   System.out.println("even");
else
   System.out.println("odd");
```

More examples and more explanation can be found here: https://www.sitepoint.com/java-ternary-operator/

Ternary conditionals almost always lower the readability of a program so in general, I prefer that you use if-else statement.