G – Raising Exceptions

[DRAFT - ~95% Complete]

# LOG Review

These are some very brief points regarding the concepts and skills listed in the LOGs for this topic.

**General Programming Concepts and Terms**

* **Describe how object-oriented programs run.**
  + A computer program is a set of instructions for manipulating information.
  + Every computer program has a single starting (or entry) point called the Main method. The main method is responsible to “run” (or “execute) the program, and when the main method exits, the program closes and stops “running”. The length of time that a program is running is called its “lifetime”. As a program runs, it executes instructions in the program’s code, one at a time; this is called the “flow of execution”.
  + During the lifetime of a program, the main method creates objects and calls methods on those objects. Each method is basically a set of instructions to be executed. Those methods can, in turn, create other objects and call other methods.
  + Whenever a method is called, the operating system keeps track of which method is calling another method through something known as a “call stack”.
* **Describe what is meant by a “call stack”.**
  + A call stack is a mechanism to track which method is calling another method while the program is running. Call stacks allow the computer to know where to “return to” when a method exits.
  + At any given point during the program’s execution, the method that is currently executing is referred to as the “called method”. When the called method exits, it returns control back to the “calling method”, thereby allowing the calling method to continue its own execution.
  + Whenever a method calls another method, the “calling method” pauses and waits for the “called method” to finish; once the called method returns, the calling method will resume executing its own code.
* **Define the term “Exception”**
  + An exception is a message of some run-time error that has occurred in a program. Exceptions interrupt the normal flow of execution in methods, forcing the method to immediately exit. Methods generate exceptions by using the “throw” statement. Any method may generate an exception (including constructors).
  + The throw statement exits the method by sending back an object (called an “Exception Object”, or simply an “Exception”) that can be used to identify the kind of error that has occurred. The calling method then has the opportunity to deal with (or “handle”) the error; if the calling method does not handle the error, then the exception continues to go back through the call stack until it finds a method that will handle the exception.
  + If none of the methods on the call stack handle the exception that was thrown, control is eventually returned to the main method. If the main method does not handle the exception, then the exception is sent to the operating system. At that point, the operating system shuts down the program in what is called an “abnormal program termination”.
* **Describe what is meant by the phrase “raising (or throwing) an exception”**
  + To “raise an exception” (or “throw an exception”) is to force a method to exit prematurely by sending back an “Exception object”.
* **Describe the role that exceptions play in a computer program**
  + An exception is an interruption in the normal flow of execution in a method; exceptions force the method to immediately exit.
  + Exceptions provide a way for methods (including constructors) to report errors to whatever code that is calling the method. These errors are usually (but not always) due to invalid information passed into the methods via the parameters. Through the use of exceptions, methods can perform validation on their parameters and report any errors that might prevent the method from being able to perform its task.
* **Identify the three most commonly used Exception types used in this course.**
  + .NET Framework
    - **ArgumentException** – Typically thrown whenever an argument passed to a parameter does not have an “acceptable” value.  
      For example, if a method has a parameter called “percent”, it might perform validation to ensure that the value is not less than zero or greater than 100. If the value is not valid, it may be appropriate to throw an **ArgumentException**.
    - **ArgumentNullException** – Typically thrown whenever an object is checked for and found to be “null”.   
      *(***ArgumentNullException** *will be discussed in the topic on Enumerations and Composition.)*
    - **Exception** – This represents a general exception, and can be used whenever there is no other suitable exception type.

**OOP Basics**

* **Explain why methods (including constructors) should perform validation of their parameter values**
  + One of the principles of object-oriented design is that objects must be responsible for ensuring that their fields have correct values. This is known as ensuring a “valid state” for an object.
  + Typically, the values that are stored in fields come in through the parameters of a constructor or a method. Because objects are responsible for ensuring a valid state, the class’ methods must check all parameters that might allow “invalid” data to be passed in.
* **Explain the significance of raising an exception in the constructor**
  + If an exception is thrown from a constructor (or passes through it on its way up the call stack), the object’s construction is interrupted. Because the construction of that object is interrupted, the object does not get instantiated.
* **Use exceptions as part of the validation of parameter values in methods and constructors.**
  + *Demonstrate your ability to meet this learning outcome guide by completing the related assignments.*
* **Explain why property setters are a good place for throwing exceptions.**
  + Generally speaking, the only method that should directly modify a field is the property setter associated with the field. Because of this principle, the property setter is the ideal place to perform validation and to throw exceptions. When validation takes place here, it eliminates the need for duplicating the validation elsewhere in the class.
  + Validating input and throwing exceptions from within setters is a good way to follow the ***DRY*** principle: “**D**on’t **R**epeat **Y**ourself”.
* **Identify when constructors should throw exceptions directly.**
  + Whenever a field has a property setter that performs validation, the constructor should use that setter to store information in the field. If no property setter is available to process the parameter values sent through the constructor, then the constructor should perform its own validation on those parameters.