

**CIS 345 – Business Information Systems Development II – Fall 2014**

**In-class exercise: Isolating Numeric User Input and FormatException Handling**

*(and making your code cleaner!)*

**Due on Blackboard: Tomorrow, Tuesday, November 3 by 10 PM**

Exception handling within C# is not complex. A try block encloses some C# statements, which throw exceptions. A catch block catches the exception, which prevents the program crashing.

However, what happens next is what makes exception handling cumbersome. Since you want to return program control to the original point of termination, code starts getting messy with the introduction of various looping structures. None of this has to do with implementing exception handling. But rather it is to do with what happens after the exception handling. Further, the entire structure of messy code needs to be repeated every time you catch an exception.

The exceptions that are thrown because of formatting errors, such as entering alphanumeric input for an integer, can largely be managed quite easily by isolating that code into one method and calling the method whenever we want to read an integer from the console. *One should do this for all repetitive code.*

So, one part of this in-class exercise is to implement a ReadInteger method that isolates all the try-catch and looping structures to solve the problem of messy try-catch handling. The other part of this exercise is to do this in a new structure and in a new type of Visual Studio project.

Once we write reusable methods, we need a mechanism to distribute them. Methods are contained in classes. Classes can be distributed as .cs files. However, Windows includes the ability to have classes available in binary format in the form of Dynamic Link Libraries (DLL). A DLL file merely has classes compiled into it. So, for this exercise, house your class in a DLL file, which can then be loaded into a new project.

## ***Project Utility Library***

1. Create a new Visual Studio Project for a Class Library (make sure you do NOT create a console application project). Call your project UtilityLibrary.
2. Within this project, add a class, Utilities.

### ***Class: Utilities.cs***

3. Within the Utilities class in the UtilityLibrary project, add a method called ReadInteger as specified below. Make sure you read how it method is going to be used before you start writing the method. ***Detailed instructions on implementation are on the next page.***

The ReadInteger method is going to replace two lines of code that are typically written when you want to read an integer from the console.

Line 1: Console.Write("Enter score 1 for Michael: ");

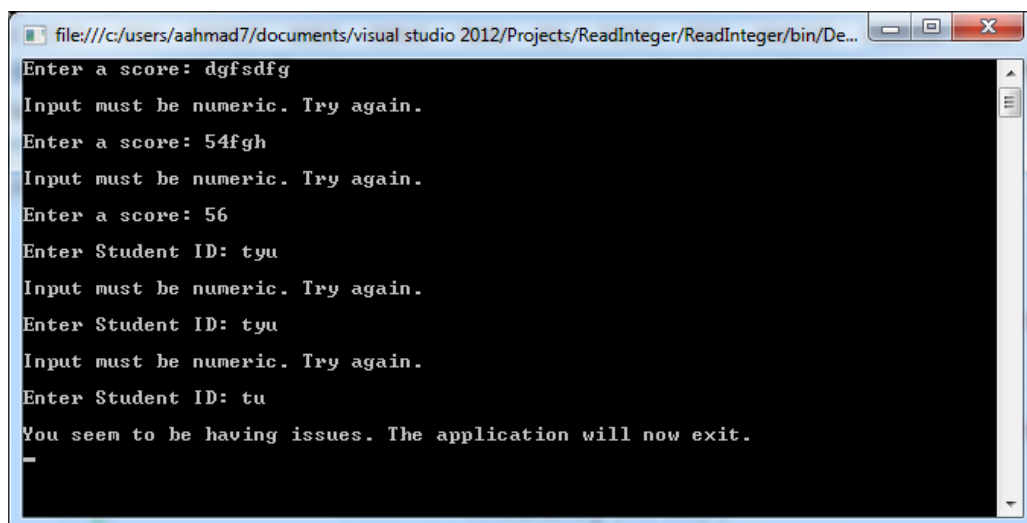
Line 2: int score1 = Convert.ToInt32 (Console.ReadLine());

The two lines above will be replaced by the following:

```
int score1 = Utilities.ReadInteger("Enter score 1 for Michael");
```

In the new code seen above, you are able to call the static method ReadInteger and pass to it a string message. The string message is what you would have written to the Console using a WriteLine statement. The ReadInteger method will display that message to the user, do a ReadLine(), convert the string to an integer and return the integer. In addition, it will handle all formatting errors and if there is an error, it will prompt the user to enter a number again and will display the message again. It will give the user 3 tries. After the third bad input, the ReadInteger() method will print a good-bye message and cleanly exit the application.

## ***Sample Output***



```
file:///c:/users/aahmad7/documents/visual studio 2012/Projects/ReadInteger/ReadInteger/bin/De...
Enter a score: dgfsdfg
Input must be numeric. Try again.
Enter a score: 54fgh
Input must be numeric. Try again.
Enter a score: 56
Enter Student ID: tyu
Input must be numeric. Try again.
Enter Student ID: tyu
Input must be numeric. Try again.
Enter Student ID: tu
You seem to be having issues. The application will now exit.
_
```

*Method Name: ReadInteger*

*Method Type: Static*

*Purpose: ReadInteger prompts the user to enter one integer. If the integer is not valid, it will handle the formatting exception and prompt the user again. It will try three times. If successful it will return an integer. If not, it will exit the program.*

*Access Modifier: Public*

*Parameters: a string called “message”, which the developer wants displayed to the user when he or she is prompted to enter a number.*

*Return type: integer*

*General Method logic:*

Declare an integer called numberOfErrors. Initialize to zero.

Declare an integer called number. *This will store the number you read.* Initialize to zero.

Declare a boolean called promptUserAgain. Initialize to false. *This will determine if you want to prompt the user again to try entering the number again..*

Do

Start a try block

Display the message to the user. *The “message” is in the parameter variable.*

Read an integer. Assign it to number.

*This line executes only if the previous one didn't have an error. So,*  
set promptUserAgain to false.

End try

Catch Block – catch FormatException

If the number of errors is 2, *User has already made 2 errors. This is the third error!*

Tell the user they are having problems entering data and that they should  
press a key to exit

ReadKey or ReadLine

Use Sytem.Environment.Exit(0) to exit the program.

End if

*You only get to this line if the number of errors was under 2. So,*

Tell the user that input must be numeric

Set promptUserAgain to true. *There was an error, prompt user again!*

Increment the variable that stores the number of errors.

End Catch

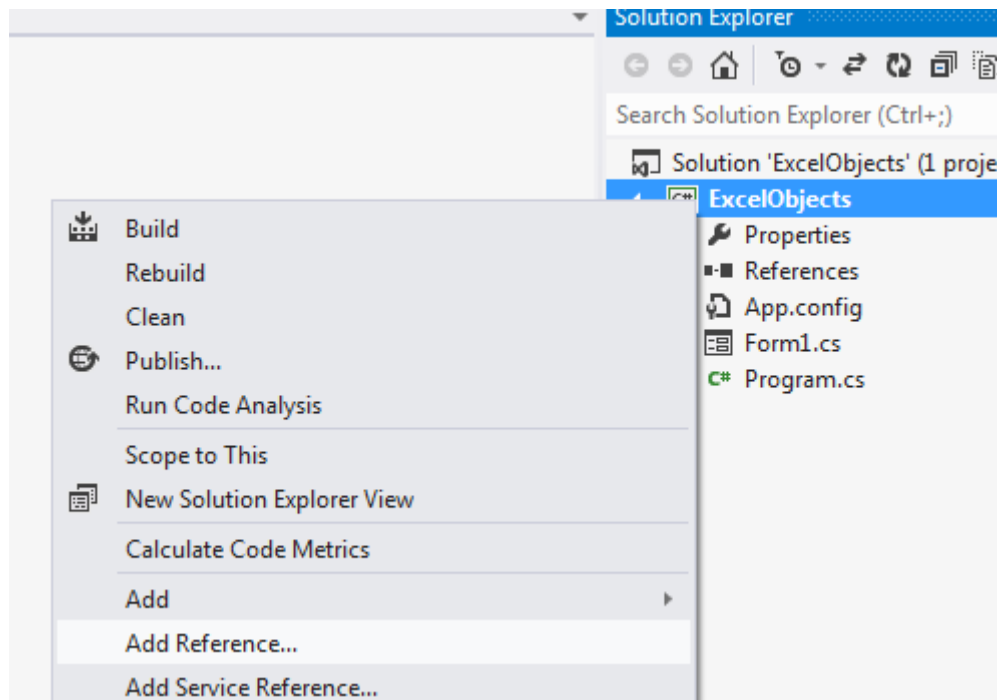
While promptUserAgain is true (*this is the end of the Do loop*).

Return number.

4. After you are done writing the method, go to the Build menu and build the project. The status message in the status bar below should say “build succeeded.” However, there will be no other output. *There is no program, there is no main, no method is being called, so nothing runs and there is no output!*
  - a. Close this Visual Studio Solution.
  - b. Go to Windows Explorer and find your project folder. Go into its bin\debug directory and you should find a UtilityLibrary.dll file. Copy that file.

***Project: DLLTestApp***

5. Create a new project of type “Console Application.” Name it DLLTestApp. *Make sure you check that you are creating a Console Application!*
  - a. Copy the DLL from the earlier folder to the project folder for DLLTestApp. Put the file in the bin\debug folder.
6. Add the DLL file to the project as a “Reference.” To do this, right-click the project within Solution Explorer, and click Add Reference.



- Click Browse and select the DLL file.
- Click References in Solution Explorer and verify that it shows your DLL project. If it does, then adding the reference was successful.

4) Tell the compiler that you want to utilize the namespace from the DLL by add a “using” statement at the top of your class.

- If your namespace was UtilityLibrary, then you would use that name after “using.” *Using statements are placed at the top OUTSIDE of the class and namespace blocks – at the very top.*

5) After this point, you are able to use any class contained within the DLL without having access to the .cs files for those classes.

*Now, in the Main method, briefly test the behavior of your ReadInteger Method.*

*Method Name: Main*

*Purpose: Utilize your newly written ReadInteger method to read integers.*

- Declare an integer, scores and call ReadInteger to read it. ReadInteger expects a string argument with a message. Provide it with the message “Enter a score” as a string. Assign the return value of ReadInteger to scores. *All this is one line of code.*
- Declare an integer, studentID and call ReadInteger to read it. ReadInteger expects a string argument with a message. Provide it the message “Enter a Student ID” as a string. *Remember that integers **cannot** be 9 digits long! For that, you will have to use other bigger data types.*

*Examples: int choice = Utilities.ReadInteger ("Select menu options 1-3: ");  
int studentID = Utilities.ReadInteger("Enter student ID");*

---

Your programs will now be able to call ReadInteger rather than invoking Console.ReadLine and Convert.ToInt32 throughout the program. This will provide an easy way to read numbers and have exception handling implemented without having repetitive try-catch statements.

Copy your Utilities class in every project you want to use it in! You may copy .cs files directly or load the compiled binary DLL file.

### Optional Enhancements.

- Create ReadDouble and ReadString methods for future use in your programs.  
*ReadString() does not need any data type conversion or any try-catch handling. However, you should do it for consistency. Then ALL of your user input will be done using your own custom Utilities class in the same format.*
- Create an enhanced ReadInteger method.

An enhanced ReadInteger could accept additional parameters that take in the minimum and maximum values that should be considered valid input. Thus, ReadInteger could do validation and prompt the users to enter a number again if it wasn't in the valid range *even if* there was no formatting error in there.

The method call might look like this:

```
intChoice = Utilities.ReadInteger ("Select menu options 1-3: ", 1, 3);  
int intPosition = Utilities.ReadInteger ("Enter a value from 0-9: ", 0, 9);
```

- Modify a substantial project, such as the Assignment 3 application to use ReadInteger

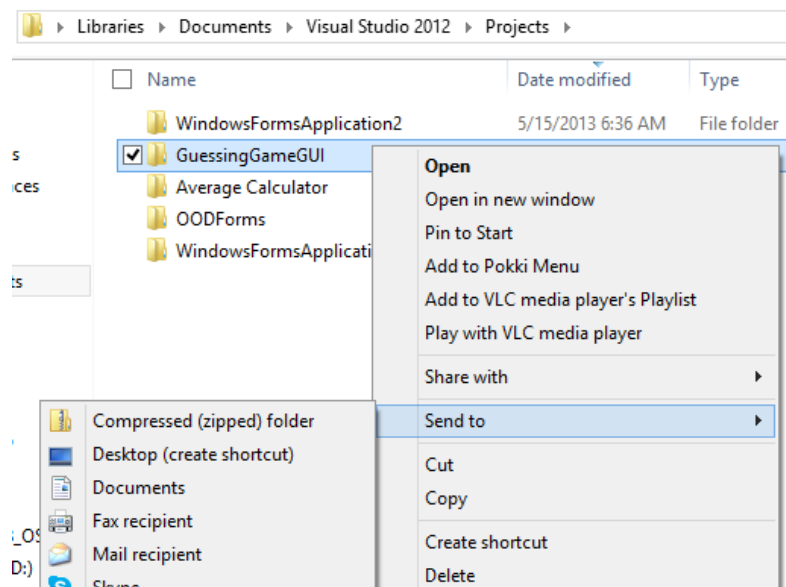
## **Submission Instructions**

Submission should be made using a zip file that contains all of the Visual Studio C# project files. You will need to **zip the entire project folder** along with the .sln and .suo files. The folder will automatically contain the class source files as well as the executable file that is generated in \ProjectName\bin\Debug folder. Upload file to the Blackboard assignment drop box.

Make sure you check the following. Your grade is dependent on all these criteria being met.

- You have included your name as a comment within your class.
  - e.g. “ // Inclass 10, Jane C. Smith, CIS 345, Tuesday 1:30 PM”
- **Submit your UtilitiesLibrary project folder (the one in which you wrote the DLL).**
- Zip filename is: Inclass10.zip

Zip the entire top-level folder by right-clicking the folder and selecting Send to | Compressed (zipped) folder.



Using built-in windows zip tools: <http://windows.microsoft.com/en-US/windows-vista/Compress-and-uncompress-files-zip-files/>