Ryan Biswas

Oakton HS FBLA

Desktop Application Development 2015

**Technical Specs**

100% Visual Studio 2013 Express

C# language

150+ hours

REQUIRES MICROSOFT 4.5 .NET framework and Windows 7 or higher

**Reference**

I started out this project with very little knowledge of C#, however the syntax of C# is very similar to Java, which I have had experience with from AP CS. The major difference between C# and Java were the libraries and new functions. I was able to learn on the go thanks to these three websites:

Official Microsoft API guide – main reference

<https://msdn.microsoft.com/en-us/library/ms123401.aspx>

Stack Overflow– specific questions

[www.stackoverflow.com](http://www.stackoverflow.com)

DotNetPerls – Broad Tutorials on C# concepts

<http://www.dotnetperls.com/>

**Instructions**

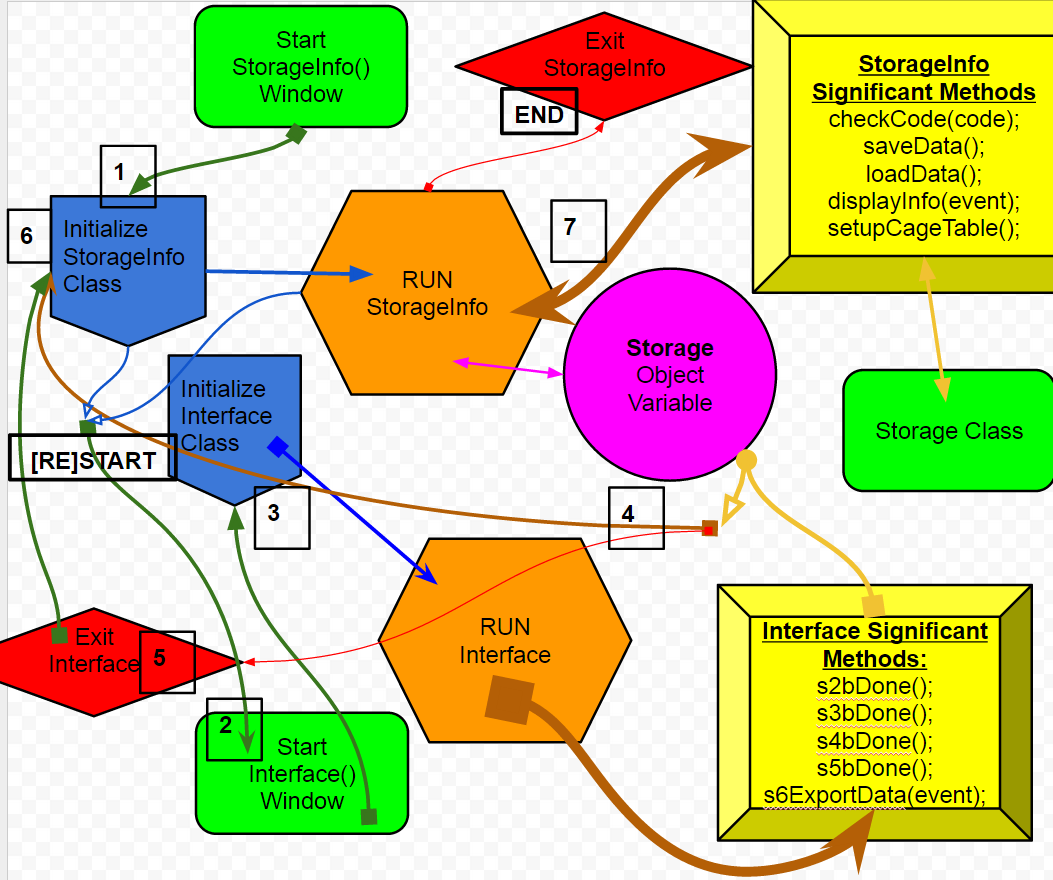
This program features a large UI for easier operation. This program comes in a package with the app itself, 3 C# class files, 1 appData file.

1. To get started unzip all the contents into ‘C:\NLCDesktop\’
2. Open ‘Ryan Biswas FBLA.exe’
3. Then the program will take over from that point and provide instructions on how to advance the prompt.
4. To view PROJECT, open VisualStudio 2013 and open project >”Desk…pment.sln”

**Clarifications and Important Information.**

1. The Expense report can be found by clicking the STATISTICS button, contains total cost of each category. Animal vs Shelter information is found by clicking GRAPH button.
2. The Case ID is like a username, which is unique to the animal and can be found by searching up. Also must enter a unique ID when adding a new animal; header >> Green
3. The Inventory report can be found by clicking the header on the animal list (left most object in StorageInfo) and then cycling through the configurations.
4. Search for a specific animal by using the Case ID
5. There is a tool tip help popup for almost EVERYTHING.
6. In order to progress screen on Interface window, make sure ALL the field headers are green.
7. Interface application can be entirely progressed by using the tab key (in order too)
8. Click on RESET tab to completely restart the application process. Use the previous tabs to make changes to input fields (MUST CLICK DONE ON THAT TAB).
9. You can click on the suggested Case ID results dropdown to automatically fill in the field.

**Application Structure Guide: (Vast Generalization)**



Key

Arrows represents the path of the program.

Arrows are colored based on its purpose and root.

Arrows that point to a block represent a call to the block’s methods

Arrows that end with a ■ represent exit/progress, the program will move to a different method(s) and won’t transverse this function again

Arrows that end with an another arrow head, ► ,resemble back and forth method calling

Arrows that have an outlined head » represents split/join where the arrow carries the function’s motives in two different directions

Arrows that have no base, represent just normal method calls

Arrows that have a circular base, • ,are functions that call multiple other functions on its way.

**Special Noteworthy Program Features**

* **Save and Load Data**: This program can save the entire array of Storage by converting each item into a string and adding them into a .data file with special formatting. Then the program can read each line of the .data file and reconvert it back into usable data.
* **Statistics**: The program can compute data and display the totals, averages, and most popular.
* **Graphs**: The program can convert statistics and display it as Pie Graphs.
* **Help**: Almost every component in the program has an associated HelpTip, which provides instructions when hovered over by the mouse
* **Error Management**: Program checks over interaction and displays a message box to indicate error instead of crashing.
* **Commentary**: Almost every function and command has extensive commentary/explanation
* **Universal Naming Conventions**: Almost all methods & variables follow a standard, easy to understand, naming system.

**User Convenience Functions (U.C.F):**

Small set of methods that smoothen the flow of program and ensure that data entered by the user follow the correct format. They also make the program more dynamic. They are mostly event methods, meaning that they are raised when a user interacts with the Control. So therefore they are part of the RUNTIME group and exist in both Interface and StorageInfo classes. They are effective enough to delude the user of its existence and function.

* **Tab Control:** When user is done with the current screen, a new tab is added based on what the user entered. This maintains a chronological flow to the program and can be back tracked to change entered information.
* **Reset Input:** If user wants to restart the whole process, an unchanging tab at the top left corner can be clicked to clear ALL entered data and variables and reset any user interaction.
* **Watermark Effect:** All TextBoxes and ComboBoxes display a preliminary instruction text in their input field. This text is automatically cleared when the user focuses on the input field. However if the user doesn’t enter anything the field resets back to its Watermark text.
* **Numeric Input:** A special attribute that only allows the user to input numbers.
* **Progression Marker**: If the watermark is removed from an input field, it changes its color to bright green to indicate progress. This attribute is checked at the end to allow the user to progress onto the next screen.
* **Additional Information**: The fields for additional information are only revealed if the user prompts it by interacting with a field that requires it.
* **ComboBoxes**: User can either type information into the ComboBox or select one of the preset options already available by clicking ANYWHERE on the field.