Name of program.

Encrypted rendering encoded timed events.

Record Encrypted data to a file

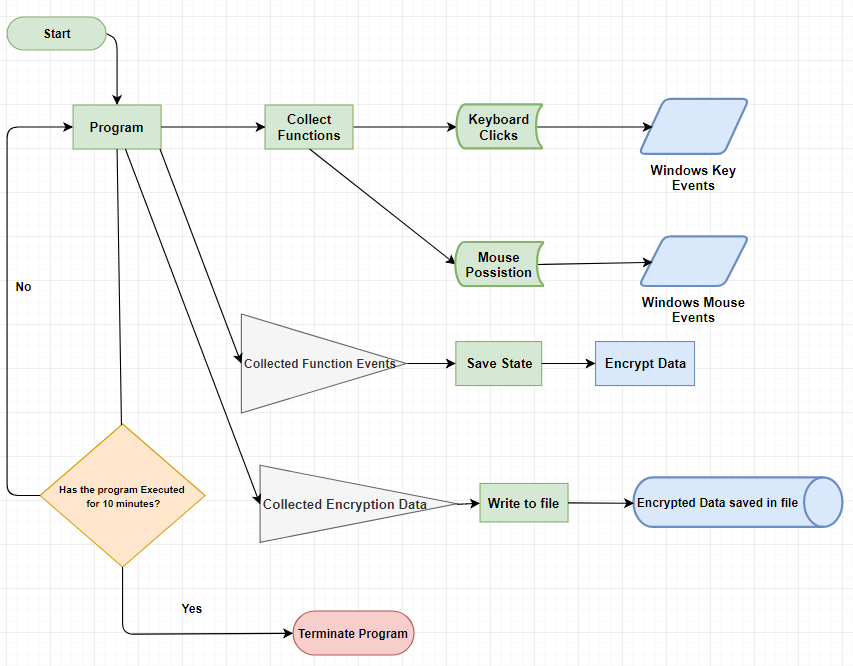
(Name of program) was created to demonstrate and apply my computer science related skill. The program I developed records in a file encrypted data. The data recorded are windows event based, specifically windows keys and the mouse position. Coupled with the keys and mouse position is a timer to keep track of when an event is raised. When the events and time are received, they are then sent through an encryption algorithm and written to a file in a predetermined location.

This project furthered my understanding and knowledge around how C#, Visual Studio, and the .net framework interacts and gives instruction to computer components. To further extend my understanding and knowledge, I also applied forms of encryption to my (Name of program).

What languages or frameworks are included?

The Computer language that I used was C#, I used the .NET Framework to compile the code written.

To introduce you to how I wrote the program I have included a flow chart to expand upon what is happening, and how the computer program works.



Green - Static Functions listening.  
Blue – Events raised and recorded.  
Grey – Event and non-event data collected.  
Orange – Condition for program termination.  
Red – Termination of program.

Proceeding is a general overview of the step by step guide to how the program works.

**Program:**

When the program is executed (Start), it goes through various functions in program. These functions are of a static type nature.

**Collect Function:**

Program talks to collect function to collect two pieces of information, the key that is pressed and the mouse position on the screen.

**Program:**

Once the Collect function performs its tasks, we return those tasks to program and report to a function called Save State.

**Save State:**

Save State then gathers and adds more data in preparation for an encryption algorithm. We then send the collective data into and encryption function.

**Program:**

We return that encrypted data and send it to another function to write to a file.

**Write to file:**

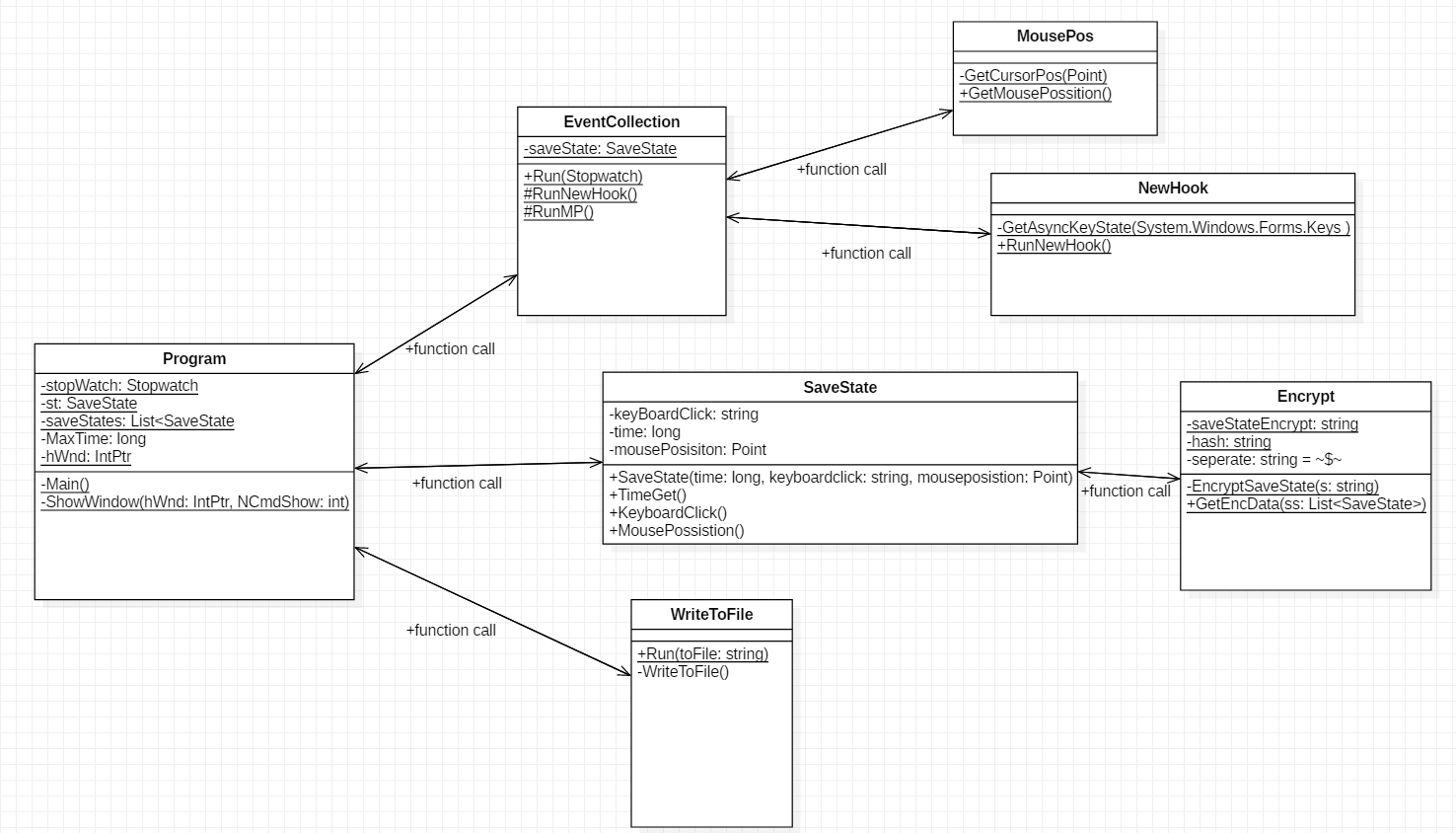
We write encrypted data to a file located within the program location.

**Program:**

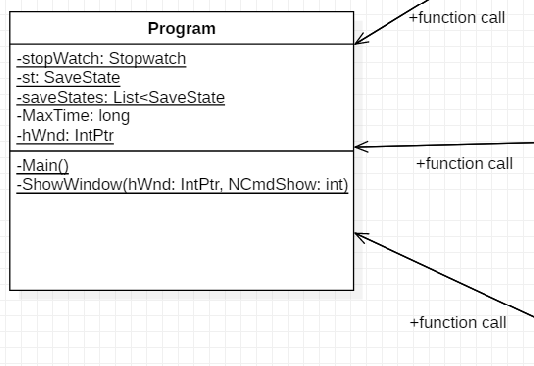
The program checks to see conditions whether to exit the program or continue. The termination of the program is determined whether it has been ten minutes or not. If it has been ten minutes, the application closes.

The contents in the file consist of appended strings. Each string is an encrypted time, keypress, an x mouse position, and a y mouse position. Since the string went through the encryption process the human when glancing upon the file will only see a random amount of number and letters.

I have provided an in-depth analysis of how the program is run through the use of a Unified Modeling Language (UML). With the help of the UML we can start to understand the how the program works instead of what it does. I have included the functions that perform the desired tasks and show the variables that each task uses to complete the application process.



When we look at the UML we notice similarities between it and the flow chart above. I will describe through similar steps as the flow chart using the UML. I have include pictures to view at a closer look at how each class is prepared and show the OOP operations performed.



**Program Class:** Start of the program.

**Variable:**

In this class we see a few variables, stopWatch, st, saveStates, MaxTime, and hWnd.

***stopWatch:*** is of type Stopwatch. We use this variable to determin the time of the button click and to determin the end of the program.

***St:***is a varaiable of type SaveState. St is used to store data that will be retrieved from main funciton.

***saveStates****:* is a variable for a C# List of type SaveState, this will store a lists of SaveState objects so that a future function can have access to the desired varable in the variable called *st*.

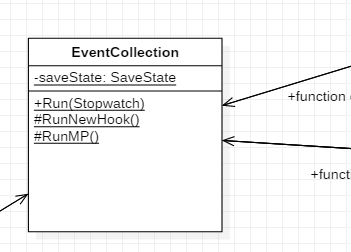
***MaxTime:***is of type long. Maxtime is the variable that limits the program to run for a set amount of time.

***hWnd****:* this variable is of type IntPtr, this type is specificly made to be for pointing in memory. We use this pointer to assign *hWnd* with a specific number.

**Funcitons:**

**Main():** Main function is where the program starts. Inside this function is where all procedures occur. When a procedure is finished a return type is then assigned into the types that are withon the main function.

**ShowWindow(IntPtr, int):** ShowWindow function is a function call that requires a type pointer such as IntPtr, and an integer. When the function is called the purpouse for this application is to make the console transperant and not on the screen.



**EventCollection Class:** Function called from main function.

***saveState:*** This variable is to be created of type SaveState to create an object that holds a long, string, and point.

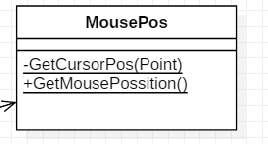
**Functions:**

**Run(Stopwatch):** Run is a static function that passes the stopwatch so the program has a numericle way to orginise what keys and what possition is at that moment in time.

**RunNewHook():** RunNewHook is a static function to retrieve a string of a key whether a key is pressed or not. If a key is not pressed in this pass of the function call, the default ascii key is 00.

**RunMP() :** RunMP is a static function to retrieve a Point. This point is what makes it so that I am able to get the X and Y posistion of the cursor that is on the screen.

After the retrieval of these Functions they are then relay to Program Main function.

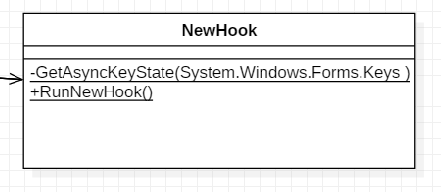


**MousePos Class:** This class is called from the EventCollection Class

**Functions:**

**GetCursorPos(point):** GetCursorPos is a private function designed to retrieve a Point from the low level languages from the windows environment.

**GetMousePossition():** GetMousePossistion is made so that the private function is accessible to whoever calls for a mouse posistion.

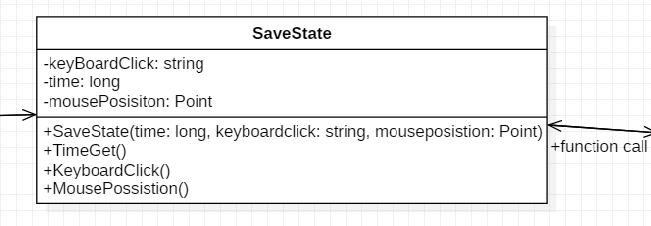


**NewHook Class:** This class is called from the EvenCollection Class

**Functions:**

**GetAsyncKeyState(Key):** Private function to retrieve the key from low level windows environment.

**RunNewHook():**  RunNewHook is a static function when called will return the string that GetAsyncKeyState function collects.



**SaveState Class:** This class is a container for collected data that the function calls are collecting.

**Variable:**

**keyBoardClick:** A variable that represents the ascii character of what was clicked from an oustide function call.

**time:**  Long variable to collect the exact time that the key is pressed and the mouse posistion.

**mousePosistion:** Is of type Point, for the use of the X posistion property and the y posistion property.

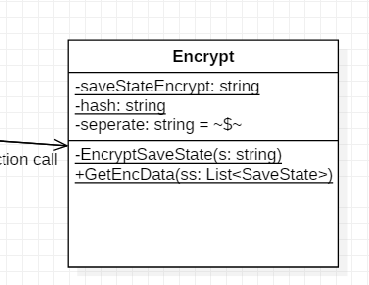
**Function:**

**SaveState(long, string, Point):** When this function is called it is to retrieve and store the variables such that each time it is called a new object with assigned propertys.

**TimeGet():** Getter when called gets the long varable to this object

**KeyboardClick():** Getter when called gets the string varable to this object

**MousePossistion():** Getter when called gets the Point type to this object.



**Encryption Class:** Called from main function when a string is passed thorugh. The classes functions and design are to encyrypt a string to MD5 encryption from the windows encryption library.

**Variables:**

**saveStateEncrypt:** saveStateEncrypt is a varaible of a new string that is assigned within the classes function.

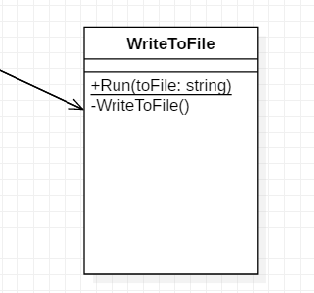
**hash:** This is a key for when funcitons inside this class are called it helps the encyption methods to remember a key so when a decrytion method is called it has a way to make sense of the string.

**separate:** separate is a constant string for future decription to know when each each property in SaveState will be recognised as one element.

**Functions:**

**EncryptSaveState(string):** Function for when the string that is past through goes through MD5 encryption and returns an encrypted string.

**GetEncData(List<SaveState>):** The program uses GetEncData functions parameter List<SaveState> and searches through all SaveState objects to retrieve the variable in SaveState and send back a single string of Encrypted Data which consist of long,string,point x, and point y.



**Write to File Class:** Writes the Encrypted string to a self made file in a self made directory.

**Functions:**

**Run(string):** Function that retrieves a string to perform write procedures in the computer.

**WriteToFile():** Function that is called from Run in WriteToFile. There is a directory that is created and a file name that is created to append the string into the file.

When code finnishes what is left is a file with 10 minutes of ecrypted data.