***Installation***

Git

GitHub

GitBash

***Week1***

***Cloning a Repository (local copy does not update established repo in GITHUB until you Commit and Push it there)***

* Obtain link for the repo
* Choose a folder to put it in (if you create one you need to reopen CMD if already open for it to see the new folder)
* Git clone <paste link> (must have .git at end)
* Then you should be able to open in Git hub and then refresh to see newest code

***CMD –***

Commands to push changes through CMD

1. git status
2. git add . (the “.” makes it git all changes in every repo the folder that you are in for the CMD)
3. git status
4. git commit -m “ <Title>” (all commits must have this) even if this is done through VS Code
5. git status
6. git branch – Tells you where it is going to publish (origin is the original repo created in GitHub)
7. git push – pushes to main branch in GitHub
8. git fetch – to see if there are changes in GitHub that are not in your local
9. git pull – brings changes from GitHub to your local if changes are made by others (need to do this for updates from trainer files to your computer)
10. code . – will open VS Code

***Command Prompt Shortcuts***

This is not user friendly but it is TECH friendly.

\*\*Best friend is your tab key

> dir (gives you a directory for the folder that you are in).- this is the less pretty version of your folders from the file folder Icon (your drives)

> cls (gives you a clear terminal back to just where you type in commands.

> cd <folder name> can put your terminal to a new folder in a folder. Ex: cd documents

> cd .. takes you back to the root folder in a terminal. (Parent directory)

> git -v this will tell you if git is installed and what version

>dotnet - - version (gives you the version of SDK you are working with)

***Git***

Git (helps manage code) and GitHub (stores versions of code) are different things.

Public vs Private settings are in the danger zone in (public means everyone in the world can see it – Private you must invite people too.

ReadMe.md will always show details on the front page in GitHub

Can manage through windows credentials but easier in CMD (--) not one dash

To change your email In CMD prompt (local/user/U713py) git config –global user.name “<name>”

To change your password In CMD prompt (local/user/U713py) git config –global user.email [rchester@geico.com](mailto:rchester@geico.com)

To see how GIT is configured git config –List

***VS Code***

Text editor that is modifiable with extensions (IDE’s –Integrated Development Environment)

Environment to code

Notepad is less powerful than VS Code

It is really light weight to run on laptops (does not take up much space/memory or processing)

C# Developer kit is disabled – must have licenses for it to be installed and cost money(WE SHOULD NOT HAVE IT)

CoPilot must be disabled – during class because our geico account is linked to our Copilot in VS Code – will be able to use after class while working --- Can we use personal emails to get different account? Does it cost money

Classes/Objects - PascalCasing - Car

methods - PascalCasing - DoSomething ()

Variables - camelCasing - numberOfBooks

Types of Data

//int - holds integer values (whole numbers only)

//double, float - hold decimal numbers

//char - holds a single character

//string - holds a series of characters (In C#, a string is treated as an array of chars)

//boolean - holds either "true" or "false"

Variables

1. declared like
   1. string name;
   2. int number;
2. Assigned a value like
   1. name = “Becky;
   2. number = 2;

Terminal Commands

1. dotnet run – runs the code that you have
2. Need to make sure you are linked to the right solution

NOTES

1. int num = 0;

If (input != null) num =int.Parse(input) ; This code changes input from a string to a number

1. string? – the ? addresses nulls in a string and clears errors.
2. Ctrl C ends the process from running in terminal after dotnet run command
3. Compiling – makes sure all programing is correct
4. Executing – actually running the code
5. Compile time and execution time are different things

***Making a new project in C#***

1. Create new folder. (try to capitalize appropriately and no space)
2. Open the terminal for that folder.
3. dotnet new console -n <folder name>
4. IF the sln (solution file) is not generated then follow below steps:
   1. dotnet new sln
   2. dotnet sln add <name of the sln file you just created>
   3. Close VS code and reopen to folder you are working on
   4. Should be good from there

***Operators***

Logical operators - return true or false booleans

(condition 1) || (condition 2) - logical OR operator, if either condition is true the result is true

(condition 1) && (condition 2) - logical AND operator, if both conditions are true, the result is true

(!condition 1) - the logical NOT operator, if this condition is true, we return false

Comparison operators - return true or false booleans

== - equal to, returns true if both values are equal

!= - NOT equal to, returns true if values aren't equal

> - greater than

< - less than

>= - greater than or equal to

<= - less than or equal to

***Conditional***

* + - 1. if (condition)

{

<do this >

}

else (or else if then else

{

<do this>

}

* + - 1. Switch (variable)

{

Case (value 1):

{

Some code

}

Case (value2)

Default case

***Arrays***

DataType[] variableName = new DataType[size];

Example

int[] numbers = new int[5];

Example of console writes

string arrString = "";

foreach (int num in numbers)

{

    arrString += num + ",";

}

System.Console.WriteLine(arrString);

arrString = arrString.Remove(arrString.Length - 1);

System.Console.WriteLine(arrString);

string[] numberStrings = new string[numbers.Length];

for (int i = 0; i < numbers.Length; i++)

{

    numberStrings[i] = numbers[i].ToString();

}

System.Console.WriteLine(string.Join(",", numberStrings));

***Methods***

* Example reverse for int
* string reverse2 = "12345";
* for (int i = reverse2.Length -1; i>=0; i--)
* {
* reverse2 += reverse2[i];
* }
* System.Console.WriteLine(reverse2);
* Example of how to reverse a string
* string reverse ="Revature";
* foreach (char c in reverse)
* {
* reverse = c + reverse;
* }

            System.Console.WriteLine(reverse);

***Main Method***

On program hit main and enter will present with template

***Debugging***

***Abstraction***

//Abstraction ---

        // intention over implemenation \*\*\*three word phrase\*\*\* focus on what it is supposed to be able to do over how it does it

        // 2 major devices that C# uses

        //  - Abstract Class (abstract methods, and concrete methods) Root class Animals then Cats Dogs Horse etc.

        //  - Interface (public methods, abstract methods)  additional label (turn animals into carinvores) (can add multiple)

        // can really choose either these days

        // Can't create an object (instaniation) they can be variables

        // Creates contractor - has to full and implementation of the thing we list

        // outlines expectations

        // from conceptional term you will here inheritance

        // Classes implement interfaces

        // boss around other classes in code

***Variables***

***Looping***

For Loop

* Best used when the number of iterations is known/Calculable by program.
* Again this will not run if the condition is not meet so print statement will not happen if condition is not true
* Benefits
  + This makes code inside of the loop simpler but you have to put more into the first line of code (header)
  + Example:
    - for(int count = 1; count <= 100; count++) //start at 1; go up to 100; add 1 each time

{

System.Console.WriteLine(count);

}

* + for (initialization; condition; update)
    - Initialization – can be declared (hard coded) (only runs once during loop then just goes between update and condition)
    - Condition – determines if runs loops and runs again and again
    - Update - the change that is made before we check condition again

While Loop

* This will keep a loop from running once a condition is met.
* Validating input is an excellent example of this.
* Checks condition before running – if condition is met it will not run
* Do not put print statement in here – if never runs will not print message that moving to next steps
* Syntax example:
  + while (num != 5)

{

System.Console.WriteLine("Please input the number : 5");

string? input = Console.ReadLine();

if (input != null) num = int.Parse(input);

if (num != 5)

{

System.Console.WriteLine("Please try again");

}

}

Do while Loop

* Runs at least one before checking condition because the condition is at the bottom (the while condition)
* Syntax
  + do

{

//whatever we want the loop to do (code)

}

while (condition);

Other Notes:

Not all CODE needs to be in the loop for example : after please try again – when they get it right it the print statement should be out of the loop as a next step

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Week 2

Fundamentals

encapsulation

    //  - Hiding or protecting our data to further control of who has access

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated+

***Branch Management (Future Topic)***

***From other class***

In this Hackathon we want to create a C# console application with the following requirments.

1 - Create a Class to model some sort of object

2 - Prompt the user for values for that object

3 - Use these values to create objects that we define (maybe use a Constructor?)

4 - We want to store these objects in a List (or Array)

5 - Handles any exceptions that may arise during the running of the application (no hard crashing)

6 - We want to be able to manage and update the values for our objects stored in the list

7 - Continues to run until the user quits the application, from within the application (no ctrl+c)

The goal of this application is to create, manipulate and modify an array of a consistent data types.

An potential example you can use is a grocery list, that a user can add items to, and then update if they have bought it or not.

<https://github.com/240415-NET/trainer-code/blob/main/Week2/hackathon2.txt>

<https://www.youtube.com/playlist?list=PLdo4fOcmZ0oULFjxrOagaERVAMbmG20Xe>