COS20007 - Object Oriented Programming

Student name: Nguyen Duc Manh
ID: 105547489

3.2P - Stack and Heap and Null Pointer

3.2P: Answer Sheet

Recall task *2.2P Counter Class* and answer the following questions.

1. How many *Counter* objects were created?

   2 Counter objects: myCounters[0], myCounters[1] (myCounters[2] is the same as myCounters[0].

2. Variables declared without the ***new*** keyword are different to the objects created using ***new***. In the ***Main*** function, what is the relationship between the variables initialized with and without the ***new*** keyword?

   - With "new": to create objects in memories
   - Without "new": like myCounters[2] is just a copy of myCounters[0] => copy existed reference

3. In the ***Main*** function, explain why the statement ***myCounters***[2].***Reset***(); also changes the value of ***myCounters***[0].

   Because "myCounters[2] = myCounters[0]" means [2] points to [0]
   => any changes to [2] will also apply to [0]

4. The difference between *heap* and *stack* is that heap holds "*dynamically allocated memory*." What does this mean? In your answer, focus on the size and lifetime of the allocations.

"Dynamically allocated memory" means memory that is allocated at runtime, with an unknown lifetime and size, and is managed on the heap. It allows for more flexible when we don't know how much memory is needed, it lives until no longer used . Unlike the stack, which is limited to short-lived since it tied to the scoped data.

5. Are objects allocated on the heap or on the stack? What about local variables?

Objects that are created by "new" in C# will allocated on the heap.
Local variables are allocated on the stack.

6. What is the meaning of the expression ***new*** *ClassName*(), where *ClassName* refers a class in your application? What is the value of this expression?

This expression will created an object (its type is ClassName) that based on that class' constructor.
The value of this expression is a pointer to the new object.

7. Consider the statement "*Counter* ***myCounter***;". What is the value of ***myCounter*** after this statement? Why?

The value of myCounter is an object of type "Counter" because that statement will call the default constructor to create the object.

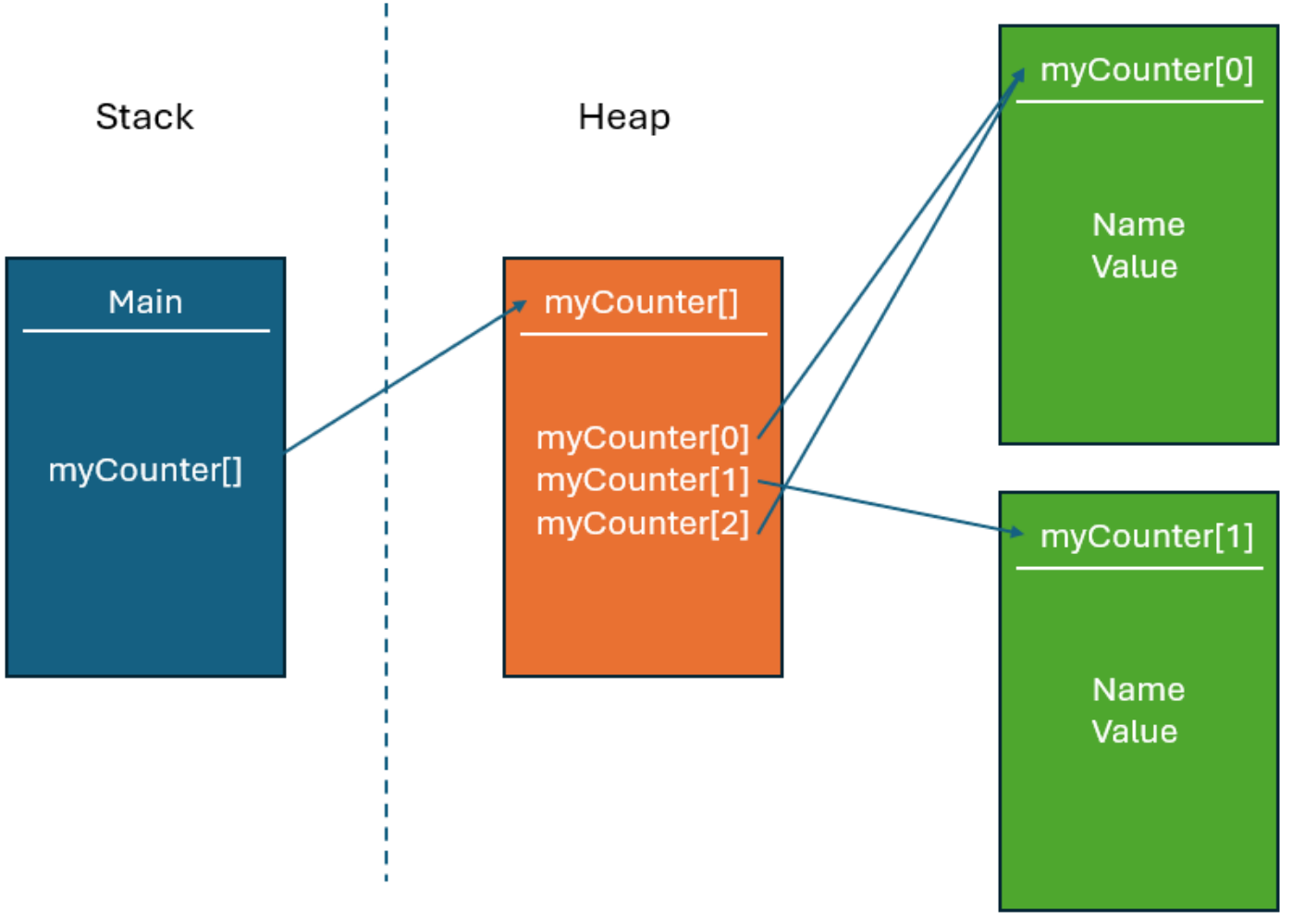
8. Based on the code you wrote in task *2.2P Counter Class*, draw a diagram showing the locations of the variables and objects in function ***Main*** and their relationships to one another.

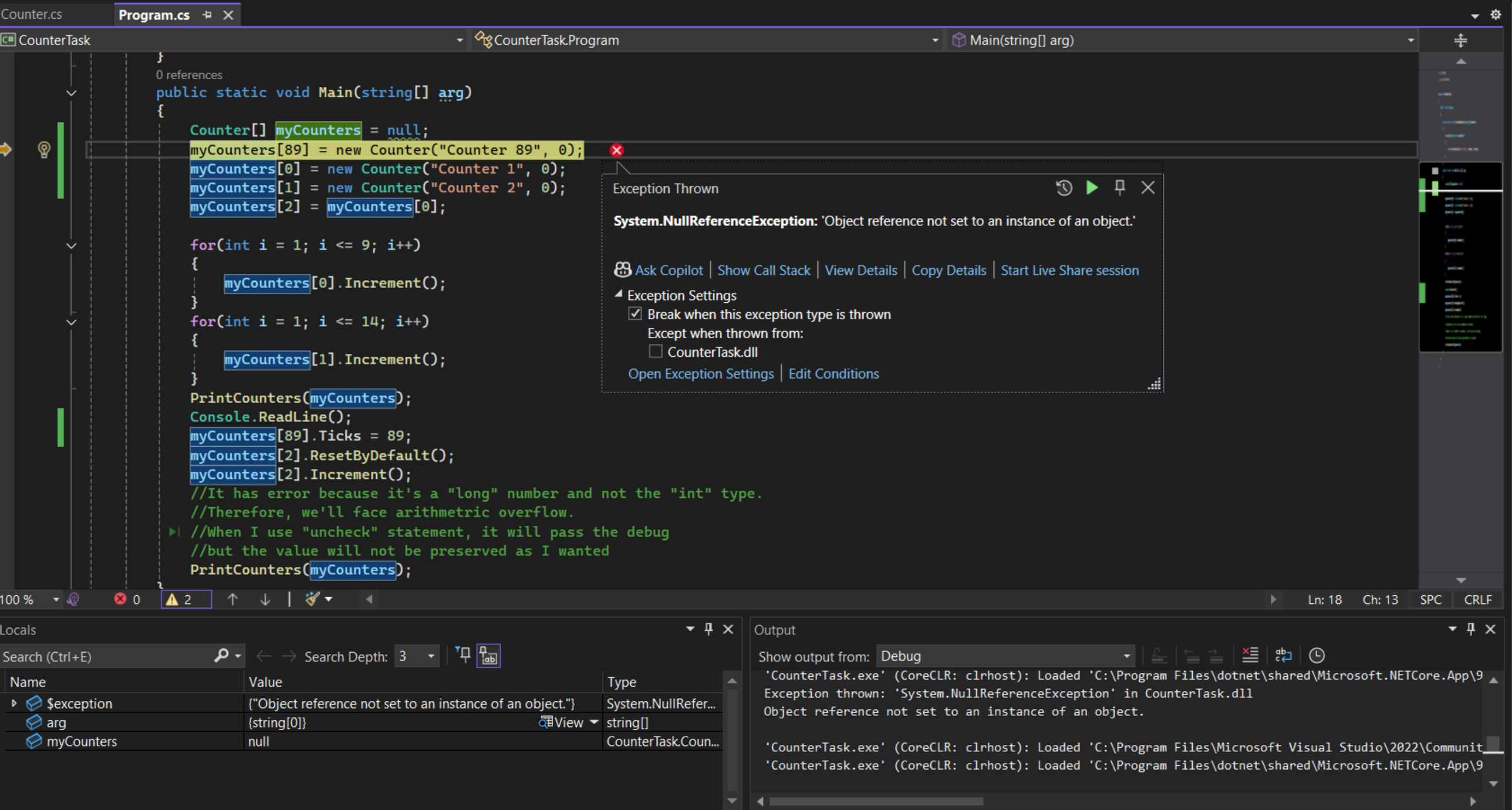
   The diagram is in the next page

9. **If the variable myCounters is assigned to null, then you want to change the value of myCounters[X], where X is the last digit of your student ID, what will happen? Please provide your observation with screenshots and explaination.**

Since myCounters is assigned to null, it will no longer reference to an array anymore.
If I try to change the value of myCounters[89], I will get the error:
NullReferenceException.
The screenshots are after the diagram.

- Null pointer CrowdStrike Bug, https://www.thestack.technology/crowstrike-null-pointer-blamed-rca/
- CrowdStrike Blog, https://www.crowdstrike.com/blog/tech-analysis-channel-file-may-contain-null-bytes/


Stack
Heap
Main
myCounter[]
myCounter[]
myCounter[0]
myCounter[1]
myCounter[2]
myCounter[0]
Name
Value
myCounter[1]
Name
Value


Counter.cs
Program.cs
CounterTask
CounterTask.Program
Main(string[] arg)
0 references
public static void Main(string[] arg)
{
Counter[] myCounters = null;
myCounters[89] = new Counter("Counter 89", 0);
myCounters[0] = new Counter("Counter 1", 0);
myCounters[1] = new Counter("Counter 2", 0);
myCounters[2] = myCounters[0];
for(int i = 1; i <= 9; i++)
{
myCounters[0].Increment();
}
for(int i = 1; i <= 14; i++)
{
myCounters[1].Increment();
}
PrintCounters(myCounters);
Console.ReadLine();
myCounters[89].Ticks = 89;
myCounters[2].ResetByDefault();
myCounters[2].Increment();
//It has error because it's a "long" number and not the "int" type.
//Therefore, we'll face arithmetric overflow.
//When I use "uncheck" statement, it will pass the debug
//but the value will not be preserved as I wanted
PrintCounters(myCounters);
Exception Thrown
System.NullReferenceException: 'Object reference not set to an instance of an object.'
Ask Copilot | Show Call Stack | View Details | Copy Details | Start Live Share session
Exception Settings
Break when this exception type is thrown
Except when thrown from:
CounterTask.dll
Open Exception Settings | Edit Conditions
100 %
0
2
Ln: 18
Ch: 13
SPC
CRLF
Locals
Search (Ctrl+E)
Search Depth: 3
Name
Value
Type
$exception
{"Object reference not set to an instance of an object."}
System.NullRefer...
arg
{string[0]}
View
string[]
myCounters
null
CounterTask.Coun...
Output
Show output from: Debug
'CounterTask.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\9
Exception thrown: 'System.NullReferenceException' in CounterTask.dll
Object reference not set to an instance of an object.
'CounterTask.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\Microsoft Visual Studio\2022\Communit
'CounterTask.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\9