3.2P: Answer Sheet

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

1. How many *Counter* objects were created?

Only 2: The "Counter 1" and "Counter 2"

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?

In the Main function, myCounters[0] and myCounters[2] both reference the same Counter object because myCounters[2] is assigned the reference of myCounters[0], meaning changes to one affect the other. In contrast, myCounters[1] refers to a separate, independent Counter object.

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

The statement myCounters[2].Reset(); also changes the value of myCounters[0] because both myCounters[2] and myCounters[0] reference the same Counter object in memory. Since they are pointing to the same object, any modification through one reference affects the other.

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" refers to memory that is allocated on the heap at runtime, where the size and lifetime of the allocation are not fixed at compile time. Unlike stack allocations, which are automatically managed and have a fixed size that is determined when a function is called (and are automatically freed when the function exits), heap allocations can have a variable size and persist until they are explicitly deallocated by the programmer.

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

Objects in C# are allocated on the heap, while local variables (such as those holding references to objects or value types like integers) are typically allocated on the stack. However, if a local variable is a reference type, the variable itself (the reference) is on the stack, but the actual object it points to is on the heap.

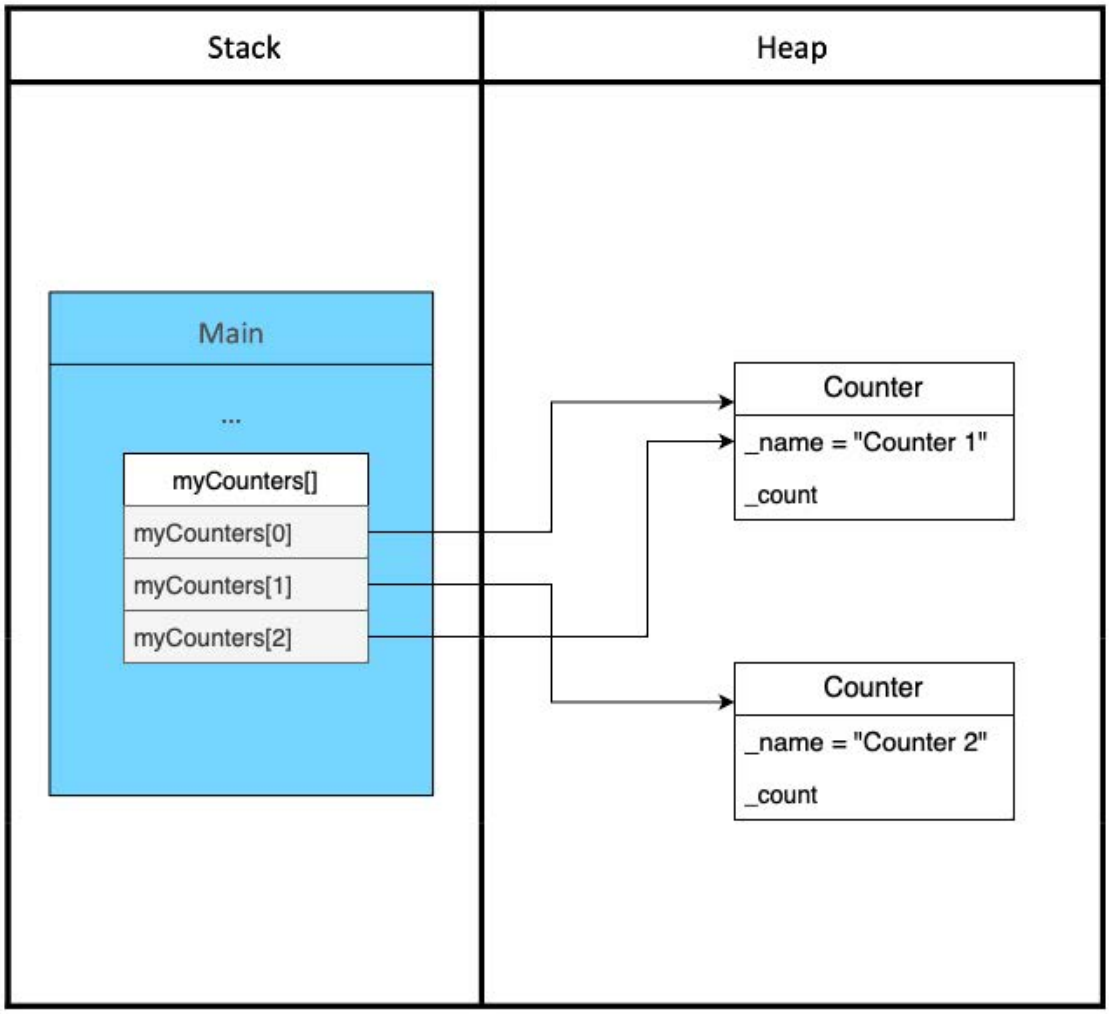
6. What is the meaning of the expression ***new*** *ClassName*(), where *ClassName* refers a class in your application? What is the value of this expression?

The expression new ClassName() in C# creates a new instance of the ClassName class by allocating memory for it on the heap and calling its constructor to initialize the object. The value of this expression is a reference to the newly created object.

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

The statement Counter myCounter; declares a variable named myCounter of type Counter, but it does not initialize it. After this statement, the value of myCounter is null because it is a reference type variable that hasn't been assigned an object yet.

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.



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

When I assign null to myCounters it no longer references a valid array, and the previously referenced array becomes eligible for garbage collection. Attempting to access or modify myCounters will throw a NullReferenceException because I'm trying to use a null reference as if it were a valid object.

```
⊗ (base) minhan6559@MacBook-Air-cua-minhan6559 Memory % dotnet run
  Counter 1 is 0
  Counter 2 is 0
  Counter 3 is 0
  Counter 4 is 0
  Counter 5 is 0
  Unhandled exception. System.NullReferenceException: Object reference not set to an instance of an object.
     at CounterTask.Program.Main(String[] args) in /Users/minhan6559/Desktop/COS20007-OOP/Week 3/3.2/Memory/Program.cs:line 26
```

Screenshot of the Main function and the ouput:

C# Program.cs 2, U × C# Counter.cs U

Week 3 > 3.2 > Memory > C# Program.cs > Program > Main

```
4       internal class Program
12          }
13
            0 references
14          static void Main(string[] args)
15          {
16              Counter[] myCounters = new Counter[5];
17              myCounters[0] = new Counter("Counter 1");
18              myCounters[1] = new Counter("Counter 2");
19              myCounters[2] = new Counter("Counter 3");
20              myCounters[3] = new Counter("Counter 4");
21              myCounters[4] = new Counter("Counter 5");
22
23              PrintCounters(myCounters);
24
25              myCounters = null;
26              myCounters[4].Name = "Minh An Nguyen - 104844794";
27
28              PrintCounters(myCounters);
29          }
30      }
31  }
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL COMMENTS

```
(base) minhan6559@MacBook-Air-cua-minhan6559 Memory % dotnet run
Counter 1 is 0
Counter 2 is 0
Counter 3 is 0
Counter 4 is 0
Counter 5 is 0
Unhandled exception. System.NullReferenceException: Object reference not set to an instance of an object.
   at CounterTask.Program.Main(String[] args) in /Users/minhan6559/Desktop/COS20007-OOP/Week 3/3.2/Memory/Program.cs:line 26
```