Task 3.2P Answer Sheet

Name: Trung Kien Nguyen
Student ID: 104053642

1. In 2.2P, how many Counter objects were created?

There are two Counter objects that were created.

2. Variables declared without the “new” keyword are different to the objects created when we call “new”. Referring to the main method in task 2.2P, what is the relationship between the variables initialised with and without the “new” keyword?

Variables declared without the "new" keyword are value types, and they are stored in the stack memory. Examples of value types include integers, floats, and booleans. When a value type is declared, memory is allocated for it automatically, and its value is directly stored in the memory location where the variable is declared.
Variables declared with the "new" keyword are reference types, and they are stored in the heap memory. Examples of reference types include classes, interfaces, and arrays. When a reference type is declared without the "new" keyword, the variable is initialized with a null value, which means it does not reference any object in memory. When a reference type is declared with the "new" keyword, memory is allocated in the heap for the object it is referencing, and the variable is initialized with a reference to that object in memory. This means that multiple variables can reference the same object, and changes made to the object through one variable will be reflected in all other variables that reference the same object.
Not all variables on the stack have to point to something on the heap. Value types are considered to be lightweight because they do not require heap memory allocation or garbage collection, and their value is directly stored in the memory location where the variable is declared.
In conclsion, the relationship between variables initialized with and without the "new" keyword is that the former refers to a reference type, which is stored in the heap memory and the variable contains a reference to the memory location of the object it is referencing, while the latter refers to a value type, which is stored in the stack memory and the value is directly stored in the memory location where the variable is declared.

3. In 2.2P, explain why resetting the counter in myCounters[2] also changed the value of the counter in myCounters[0].

Because myCounter[0] and myCounter[2] are the references to the same object.

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

On the one hand, "dynamically allocated memory" is allocated on the heap using the "new" keyword, and the size and lifetime of the allocation are determined by the program

logic. For example, in C#, when you create an instance of a class using the "new" keyword, memory is allocated on the heap for that object, and the size of the allocation is determined by the size of the object. The lifetime of the allocation is determined by the garbage collector, which periodically checks for objects on the heap that are no longer referenced by any variables, and frees the memory associated with those objects (or it will be explicitly deallocated using the "delete" keyword or equivalent mechanism in other languages likes C++).

On the other hand, when it comes to the memory on the stack, "statically allocated memory" is typically allocated on it, and the size and lifetime of the allocation are determined by the program structure. For instance, when you declare a variable of a value type such as an integer, memory is allocated on the stack for that variable, and the size of the allocation is determined by the size of the value type. The lifetime of the allocation is determined by the scope of the variable, which is typically the block of code in which the variable is declared.

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

The objects are allocated on the Heap, and the local variables are stored in the Stack.

6. What does the new() method do when called for a particular class, and what does it return?

When the new() method is called on a particular class, it will create a new instance of the object, allocate it on the Heap, store its address on the Stack, and finally returns a new reference of the object.

7. Assuming the class Counter exists in my project, if I wrote the code "Counter myCounter;" (note there is no "="), what value would myCounter have? Why?

*myCounter would have a **null** value because at that time the variable does not refer to an object in the memory.*

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