Swinburne University of Technology

COS20007 Object Oriented Programming

The Stack and Heap

PDF generated at 04:36 on Monday 21st August, 2023

Produced by OOP Submission Formatter

Task 3.2P Answer Sheet

Name: Nur E Siam
Student ID: 103842784

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

   Two. Because the value of myCounters [0] is assigned to myCounters[2].

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?

“new” generates a duplicate set of variables that can be changed with the Main code. When this is happening, the original set of variables remains constant so that if a “reset” or “new” is called, the variables will revert to default.
The array "myCounters" contains Counter objects. Initializing "myCounters[0]" and "myCounters[1]" with "new" creates separate Counter objects in distinct memory locations, while assigning "myCounters[0]" to "myCounters[2]" makes them refer to the same object. Variables without "new" are allocated on the stack, while "new" dynamically allocates memory on the heap for flexible memory management.

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

myCounters[2] directly takes its value from myCounters[0] as such, resetting myCounters[2] will also result in myCounters[0] returning to 0.

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.

The difference between memory on the heap and memory on the stack depends on "dynamically allocated memory." This involves heap memory, which is allocated during runtime and can vary in size based on user input or program conditions, providing flexibility. Memory on the heap can be allocated and deallocated as needed.

In contrast, stack memory is determined at compile-time with smaller allocation sizes and a limited lifetime tied to the duration of function calls, making it less adaptable for dynamic memory requirements.

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

Objects, generated using the 'new' keyword or dynamically allocated, typically reside on the heap, offering variable sizes and lifetimes.
In contrast, local variables declared within methods or functions are allocated on the stack, with their memory managed automatically by the compiler.

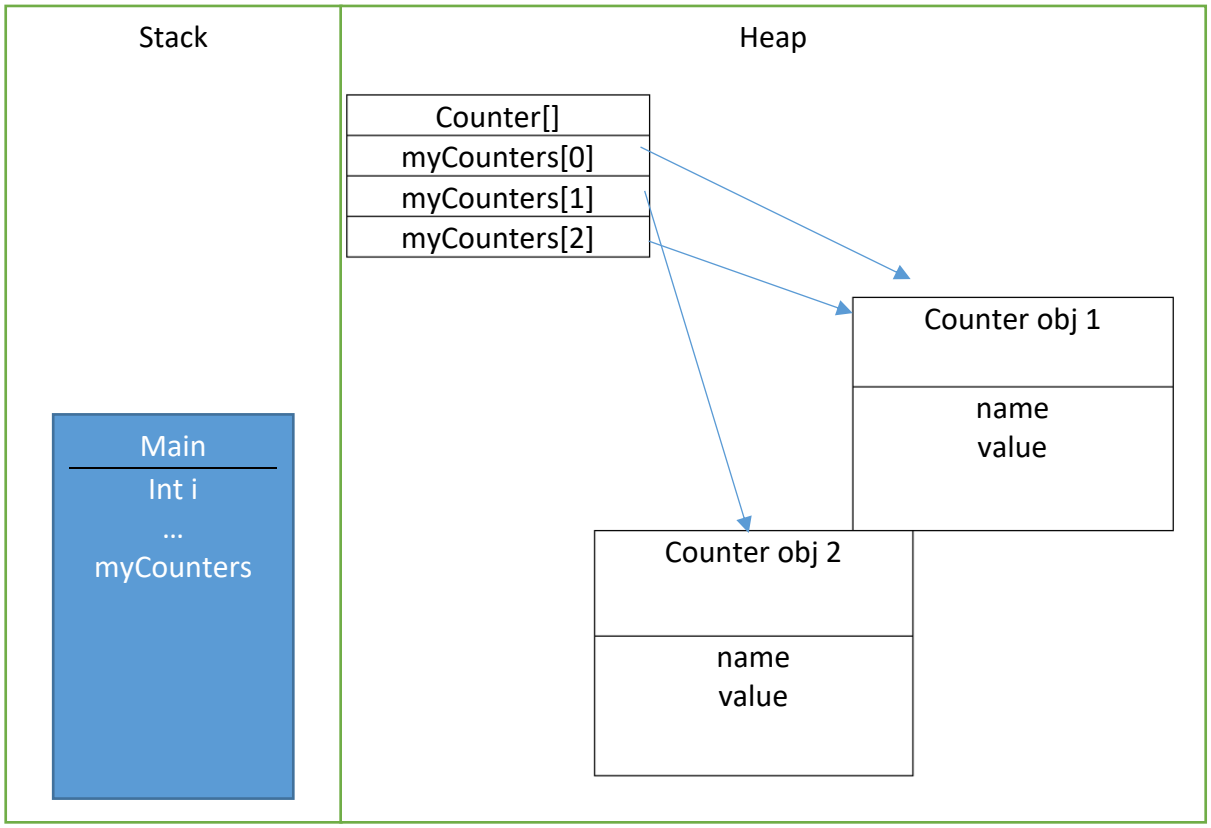
6. What does the new() method do when called for a particular class, and what does it return?

   The new() method in a class allocates memory on the heap for an object and triggers the class constructor, resulting in the creation of a new object. It then returns a reference to the object's memory location, enabling subsequent interactions and operations.

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?

The code "Counter myCounter;" would assign the variable myCounter the default "null value," indicating that it doesn't reference any object on the heap. This is because the variable remains uninitialized and lacks a specific object reference.

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.



In the main method, int ‘i’ and ‘myCounters’ are local variables that are stored in the stack.‘myCounters’ is an array for ‘Counter’ objects. The code initialises three counters (with the value of myCounters[0] being assigned to myCounters[2]) and assigns them to myCounters variables in the heap while myCounters[1] refers to separate ‘Counter’ object. Each counterobject has its own ‘name’ and ‘value’ fields. The arrows show the relationship between the objects and the variables.