These tables show (right) the order of variables declared, their addresses and types. On the right, we have the Memory's Stack. We can see that the integer was assigned the first address as 600. Following we have the double array (myArray2) starting at 608; there is a buffer between the variables. Finally, the integer array (myArray1) was declared at 634 with another buffer between myArray2 and myArray1. I do not know why 604 and 630 are empty and that would be too complex for this class anyways.

When we look at `myArray1`, we can see that the addresses are separated by 4 bytes. In our stack, each line is 4 bytes wide, so they are next to each other. When we pass in `myArray1` (or `myArray2`) into a method, we will simply use the name `myArray1`. That is because `` `myArray1` `` is synonymous to `` `&(myArray1[0])` ``. Without an index on myArray1, it is simply a pointer. Hence, we can use `*myArray1` to get the value at the zeroth index.