Memory Diagraming

This document attempts to walk through the format and generation of memory diagrams for this course. In the hardware, computer memory is a long linear array of bytes. These diagrams are to build/demonstrate an intuition for how the object instances in our programs are actually represented.

The memory diagram only represents the memory state. There are many functions that don't change the memory state (e.g. System.out.println). The main contributors to memory state changes are variable declarations and assignment statements.

Sample program/function to diagram:

```
public static void main(String[] argv) {
    int len;
    len = 3;
    int[] alpha;
    alpha = new int[l];
    alpha[0] = 15;
    alpha[1] = 7;
    alpha[2] = 21;
    System.out.println(alpha[1]);
}
```

Step 1 – Identify the variables declared in the function being analyzed. This example indicates them in bold.

```
public static void main(String[] argv) {
    int len;
    len = 3;
    int[] alpha;
    alpha = new int[len];
    alpha[0] = 15;
    alpha[1] = 7;
    alpha[2] = 21;
    System.out.println(alpha[1]);
}
```

Step 2 – Setup the variables in the stack portion of the memory diagram. In this example argv and alpha are pointers since they are not variables of a native type. The len variable is of native type.

Address	Type	Name	Value
0x000A	int	len	<unknown>
0x000B	String[]	argv	<unknown>
0x000C	int[]	alpha	<unknown>
0x000D			
0x000E			
0x000F			
0x0010			
0x0011			

Step 3 – Notice the variables that represent inputs… The values for these variables come from the caller and can't be known from the code sample we have. Note on the diagram that these values are from the caller.

Address	Type	Name	Value
0x000A	int	len	<unknown>
0x000B	String[]	argv	Received from caller
0x000C	int[]	alpha	<unknown>
0x000D			
0x000E			
0x000F			
0x0010			
0x0011			

Step 4 – Identify where new object instances are created and identify space in the memory diagram for the fields of those objects. For the array object in this example, we need space for the array length and the array elements; a total of 4 memory locations will be allocate for the **new int[len]**.

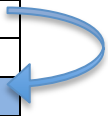
```
public static void main(String[] argv) {
    int len;
    len = 3;
    int[] alpha;
    alpha = new int[len];
    alpha[0] = 15;
    alpha[1] = 7;
    alpha[2] = 21;
    System.out.println(alpha[1]);
}
```

Address	Type	Name	Value
0x000A	int	len	<unknown>
0x000B	String[]	argv	Received from caller
0x000C	int[]	alpha	<unknown>
0x000D			
0x000E			
0x000F			
0x0010			
0x0011			

Step 5 – Assign values for native variables and link the pointers for pointer variables. Also apply function calls that might change values (none in this example).

```
public static void main(String[] argv) {
    int len;
    len = 3;
    int[] alpha;
    alpha = new int[len];
    alpha[0] = 15;
    alpha[1] = 7;
    alpha[2] = 21;
    System.out.println(alpha[1]);
}
```

Address	Type	Name	Value
0x000A	int	len	3
0x000B	String[]	argv	Received from caller
0x000C	int[]	alpha	0x000E
0x000D			
0x000E	int		length: 3
0x000F	int		15
0x0010	int		7
0x0011	int		21



Step 6 – The memory diagram is complete.