

```
1 import java.util.*;
2
3 /**
4  *
5  * Class Memory has static utility methods for allocating, clearing, and
6  * dumping an Object-valued memory array.
7  *
8  */
9 public class Memory {
10
11     /**
12      * Allocate an Object-valued memory array of the given size.
13      */
14     public static Object[] allocate(int size) {
15         return new Object[size];
16     }
17
18     /**
19      * Set all elements of the given memory to null.
20      */
21     public static void clear(Object[] mem) {
22         Arrays.fill(mem, null);
23     }
24
25     /**
26      * Dump the given number of elements in the given memory to stdout. The
27      * dump starts at memory location 0. The dump of each element is started
28      * on a separate line, prefixed with "Location n: ", for n = the index of
29      * that location. The number of lines dumped per element is dependent on
30      * the toString method for the type of element being dumped.
31      */
32     public static void dump(Object[] mem, int numElems) {
33         for (int i = 0; i < numElems; i++) {
34             System.out.println("Location " + i + ": " + mem[i]);
35         }
36     }
37
38     /**
39      * Dump the given memory to stdout, from the given startElem to endElem
40      * indices, inclusive. Note that in the "Location n: " dump prefixes, the
41      * value of n is an absolute address relative to overall location 0.
42      * Cf. dumpRelative.
43      */
44     public static void dump(Object[] mem, int startElem, int endElem) {
45         for (int i = startElem; i <= endElem; i++) {
46             System.out.println("Location " + i + ": " + mem[i]);
47         }
48     }
49
50     /**
51      * Dump the given memory to stdout, from the given startElem to endElem
52      * indices, inclusive. The difference between this method and the
53      * other three-argument version of dump is that here the "Location n: "
54      * prefixes start at 0, so this dump is relative to the given startElem
55      * beginning at 0 instead of its absolute value.
56      */
57     public static void dumpRelative(Object[] mem, int startElem, int endElem) {
58         for (int i = startElem; i <= endElem; i++) {
59             System.out.println("Location " + (i - startElem) + ": " + mem[i]);
60         }
61     }
62 }
63 }
```