Memory.java

```java
import java.util.*;

/****
 *
 * Class Memory has static utility methods for allocating, clearing, and
 * dumping an Object-valued memory array.
 *
 */
public class Memory {

    /**
     * Allocate an Object-valued memory array of the given size.
     */
    public static Object[] allocate(int size) {
        return new Object[size];
    }

    /**
     * Set all elements of the given memory to null.
     */
    public static void clear(Object[] mem) {
        Arrays.fill(mem, null);
    }

    /**
     * Dump the given number of elements in the given memory to stdout.  The
     * dump starts at memory location 0.  The dump of each element is started
     * on a separate line, prefixed with "Location n: ", for n = the index of
     * that location.  The number of lines dumped per element is dependent on
     * the toString method for the type of element being dumped.
     */
    public static void dump(Object[] mem, int numElems) {
        for (int i = 0; i < numElems; i++) {
            System.out.println("Location " + i + ": " + mem[i]);
        }
    }

    /**
     * Dump the given memory to stdout, from the given startElem to endElem
     * indices, inclusive.  Note that in the "Location n: " dump prefixes, the
     * value of n is an absolute address relative to overall location 0.
     * Cf. dumpRelative.
     */
    public static void dump(Object[] mem, int startElem, int endElem) {
        for (int i = startElem; i <= endElem; i++) {
            System.out.println("Location " + i + ": " + mem[i]);
        }
    }

    /**
     * Dump the given memory to stdout, from the given startElem to endElem
     * indices, inclusive.  The difference between this method and the
     * other three-argument version of dump is that here the "Location n: "
     * prefixes start at 0, so this dump is relative to the given startElem
     * beginning at 0 instead of its absolute value.
     */
    public static void dumpRelative(Object[] mem, int startElem, int endElem) {
        for (int i = startElem; i <= endElem; i++) {
            System.out.println("Location " + (i - startElem) + ": " + mem[i]);
        }
    }

}
```