```
                /* Storage Capacity Handling */

/* Presently, the following holds: */
%replace MAX_CLASS_NUMBER by 157;       /* Max possible is  256 */
%replace NUMBER_OF_RESOURCES by 437;    /* Max possible is 1024 */

/*
   This is the how the tables are structured:

   The 'classes' table contains one entry for each possible object class.
   Each entry contains the size of an instance of that class, the size of the
   class descriptor, a pointer to an array of resource numbers, and a
   reference count.  The reference count tells the number of objects of the
   class that have been allocated in the region.  The resources pointer
   points to a variable length array of bin(15)'s, each of which is the
   number of a resource that the class uses.  This array is terminated by a 0
   entry.

   The 'resources' table contains one entry for each possible resource
   (action, image, sound or head).  Each entry contains a size (how many
   bytes of memory that resource requires) and a reference count.  The
   reference count tells how many classes have been allocated that require
   that resource.

   What we are trying to do is keep track of the number of bytes of memory
   that the C64 is using for allocated objects.  The basic algorithm to use
   when creating an object of a given class is follows:  Add to the total
   space usage the number of bytes required for an instance descriptor of an
   object of that class and increment the class's reference counter.  If
   there are already any objects of that class, we are done.  If not, add to
   the total space usage the number of bytes required for a class descriptor
   for that class.  Then iterate through the list of resources.  For each
   resource, increment its reference count.  If it is not already present,
   add its size to the total.  To delete an object, perform exactly the same
   operation in reverse, subtracting sizes and decrementing reference counts.

   Note that there is one obvious inefficiency in these data structures which
   I left in for the sake of clarity: the resource and size entries of the
   'classes' table and the size entries of the 'resources' table are static.
   Only one copy need be kept for the whole process.  A separate set of class
   and resource reference counts must be kept for each active region.

   By my count (assuming 4-byte pointers) the static information in these
   tables occupies 2130 bytes and the dynamic information occupies 1188 bytes
   per region, given the current number of classes and resources.  Maximum
   possible sizes are 4096 bytes and 2560 bytes respectively. */

declare 1 classes(MAX_CLASS_NUMBER) init(/* Class info here */),
        1 class based(class_ptr),       /* I'm not sure of the syntax here */
          2     resources         pointer,
          2     instance_size     binary(15),
          2     class_size        binary(15),
          2     ref_count         binary(15);
declare class_ptr pointer;

declare 1 resources(NUMBER_OF_RESOURCES) init(/* Resource info here */),
        1 resource based(resource_ptr),
          2     size              binary(15),
          2     ref_count         binary(15);
declare resource_ptr pointer;
```

```
declare space_usage binary(15) init(0); /* This is the important number */

create_object: procedure(class_number);
     declare class_number binary(15);

     declare resource_array(1) based(resource_number_ptr) binary(15);
     declare resource_number_ptr pointer;
     declare a(1) based binary(15);
     declare i binary(15);

     class_ptr = classes(class_number);
     space_usage = space_usage + class.instance_size;
     class.ref_count = class.ref_count + 1;
     if (class.ref_count = 1) then do;
          space_usage = space_usage + class.class_size;
          resource_number_ptr = class.resources;
          i = 1;
          do while (resource_array(i) ^= 0);
               resource_ptr = resources->a(resource_array(i));
               i = i + 1;
               resource.ref_count = resource.ref_count + 1;
               if (resource.ref_count == 1) then
                    space_usage = space_usage + resource.size;
          end;
     end;
end create_object;

delete_object: procedure(class_number);
     declare class_number binary(15);

     declare resource_array(1) based(resource_number_ptr) binary(15);
     declare resource_number_ptr pointer;
     declare a(1) based binary(15);
     declare i binary(15);

     class_ptr = classes(class_number);
     space_usage = space_usage - class.instance_size;
     class.ref_count = class.ref_count - 1;
     if (class.ref_count = 0) then do;
          space_usage = space_usage - class.class_size;
          resource_number_ptr = class.resources;
          i = 1;
          do while (resource_array(i) ^= 0);
               resource_ptr = resources->a(resource_array(i));
               i = i - 1;
               resource.ref_count = resource.ref_count - 1;
               if (resource.ref_count == 0) then
                    space_usage = space_usage - resource.size;
          end;
     end;
end delete_object;
```