# Memory representation

As mentioned, the heap in our VM is represented as an array of unsigned integers[[1]](#footnote-2). Each object allocated on the heap has a two word header containing the size, GC bits and a class pointer. To overcome the bootstrapping problem of needing a pointer to the Class class before it has been loaded a number of “known” classes are bound to negative values during initialization and subsequently replaced when negative class pointers are encountered. The size recorded in the header is only used during GC and is not guaranteed to be the actual size of the object; see the section on our mark-sweep implementation. The pointer returned from the memory manager when allocating points to the first word after the header to make it easier for the actual object implementations to keep their offsets straight. Because the header is not part of the objects own representation it is also not part of the size given to the memory manager when allocating, and as such it would be illogical for the objects to have to take it into account when accessing their contents.

1. Technically we have a Word struct wrapping [↑](#footnote-ref-2)