# Relate and compare

We decided to do mark-and-sweep simply because we thought it would be the simplest one to implement and since we were running out of time it was important for us to actually get GC working. Doing mark-and-sweep has several disadvantages, the biggest one is probably that the entire program must be suspended during collection and no changes to the working set can be allowed. This will cause programs to freeze for a short period of time, making it incompatible with real-time and time-critical applications. In addition, the entire working memory must be examined each time. Currently our memory gets very fragmented, meaning we have to expand the memory more often that we would like. One way to solve this would be to implement a mark-compact collector so we compact the memory until all the live object are contiguous. It would not solve the problem of needing have several passes over the entire heap and we would still need to stop all threads when doing a collection.

Going forward one of the things we would like to implement are nurseries. These would be small thread-local areas of the heap where any live objects would be promoted to the next generation when they fill up. This would allow us to avoid locking the heap when allocating. Combining this with a semi-space copying collector and our current mark-and-sweep collector for a young and old generation respectively would give a solid memory manager. The semi-space copying collector is ideal for young generations as these are efficient when we have a lot of garbage, which is expected when dealing with young objects. Mark-and-sweep is, on the other hand, good when there is a small amount of garbage. Compaction may be needed in the old generation sometimes but as the amount of allocation and death is small in this part it would probably be acceptable.

We never got around to implement JIT compilation, inline caching or other techniques for dynamic dispatching so we never consider which of these techniques would suit our VM best. Our language also has classes, but since our language has a default message handler we could, without too much work, support a language without classes.