Java and Haskell memory management

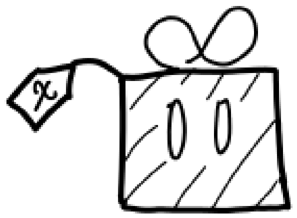
HASKELL

Haskell computations produce a lot of memory garbage - much more than conventional imperative languages. It's because data are immutable so the only way to store every next operation's result is to create new values. Every iteration of a recursive computation creates a new value. But GHC can efficiently manage garbage collection, so it's common to produce 1gb of data per second (most part of which will be garbage collected immediately). So, you may be interested to learn how GHC does such a good job.

The Haskell heap is a rather strange place. It’s not like the heap of a traditional, strictly evaluated language...

...which contains a lot of junk! (Plain old data.)

In the Haskell heap, every item is wrapped up nicely in a box: the Haskell heap is a heap of *presents* (thinks).



When you want what’s inside the present, you open it up (evaluate it).

Presents tend to have names, and sometimes when you open a present, you get a *gift card* (data constructor). Gift cards have two traits: they have a name (the Just gift card or the Right gift card), and they tell you where the rest of your presents are. There might be more than one (the tuple gift card), if you're a lucky duck!

GIFTCARD
JUST
Y
Y

But just as gift cards can lie around unused (that's how the gift card companies make money!), you don't have to redeem those presents.

Presents on the Haskell heap is rather mischievous. Some presents explode when you open them, others are haunted by ghosts that open other presents when disturbed.

JAVA

Memory management in Java:

Stack:

- In Java, the stack is used to store local variables and object references.
- Each thread in a Java program has its own stack.
- Memory on the stack is automatically allocated and freed when a function finishes executing or when an object goes out of scope.
- The stack depth in Java is limited and can be adjusted with the -Xss option of the Java interpreter.

Heap:

- The heap is used to store dynamically created objects.
- All objects in Java are stored in the heap, regardless of whether they are instance variables or local variables.
- Memory in the heap is allocated when an object is created with news and is automatically freed when there are no references to that object.
- Heap memory management in Java is managed by the garbage collector, which frees memory from objects that are no longer in use.