Object oriented programming – supervision 1

James Wood

November 24, 2014

a

1. The typical imperative language encourages mutation of variables. The typical functional language doesn't.
2. The typical imperative language doesn't treat functions as it would values. The typical functional language does.
3. The typical imperative language has built-in structures for iteration. The typical functional language relies on higher-order functions and recursion.

b

```
package code;

class S1b {
    public static void main(String[] args) {
        main(args);
    }
}
```

The `main` method calls itself immediately without any change in parameters. Logically, this puts the program into an infinite loop, but in practice, since each function call is designated a distinct area on the (finite) stack, an exeption is thrown when there is no more space on the stack.

c

If one tries to read from or write to a `null` reference, an exception is thrown. With a misplaced pointer, reading or writing will have an unforeseen effect.

d

The stack is used to store primitive values, including references. The heap is used to store objects. Only values on the stack can be directly assigned to in Java. Values on the heap need to be created via a constructor.

e

Just after `myfunction2` is called:

Stack		Heap
`x`	1	
`a`	ref0	ref0 | {1}
`num`	1	
`numarray`	ref0	

After first two lines of `myfunction2` have run:

Stack		Heap
`x`	2	
`a`	ref0	ref0 | {1}
`num`	1	
`numarray`	ref0	

`x` represents a stack value, so changing it will not affect `num`.

As `myfunction2` is about to return:

Stack		Heap
`x`	2	
`a`	ref1	ref0 | {1}
`num`	1	ref1 | {2}
`numarray`	ref0	

On the third line, `a` was assigned to a new object, and hence changes to the new object don't affect the old one.

After `myfunction2` has returned:

Stack		Heap	
`num`	1	ref0	{1}
`numarray`	ref0	ref1	{2}

`num` and `numarray` have not been affected. ref1 is ready to be garbage collected. The output is `1 1`.

f

Primitives:

- `double`
- `int`

References:

- `i` references an array
- `l` references a List
- `k` references a Double
- `t` references null
- `f` references null
- `c` references null

Classes:

- `List`
- `Double`
- `Tree`
- `Float`
- `Computer`

Objects:

- {1,2,3,4}
- `new List()`
- `new Double()`

g