A quick overview of OOP

Josh Felmeden

May 15, 2019

Contents

1 An object's Birth **1**

2 Java basics **1**
2.1 Static elements 2

3 An object's life **2**
3.1 References 2

1 An object's Birth

An object in Java is probably the most crucial piece of information (because it's called **object** oriented programming duh). An object has behaviour, attributes, and identity. An object combines both *data* and *methods* (which manipulate the data) in a single unique entity.

A **class** is like the blueprint of an object. Classes act like a module or unit of description. Here is an example class:

```
class Robot {
    String name;
    int numLegs;
    float powerLevel;

    void talk(String phrase) {
        if (powerLevel >= 1.0f) {
            System.out.println(name + " says " + phrase);
            powerLevel -= 1.0f;
        } else {
            System.out.println(name +  " is too weak to talk.");
        }
    }

    void charge(float amount) {
        System.out.println(name + " charges.");
        powerLevel += amount;
    }
}
```

A class is different to an object. Creating an object is called *instantiating* an object.

2 Java basics

Remember, a class is an immutable blueprint for an object. It has both methods and fields. An object is a stateful and unique instantiation of a class. Remember, a class is an immutable blueprint for an object. It has both methods and fields. An object is a stateful and unique instantiation of a class.

Some people don't like Java because they say that 'nouns' (objects) shouldn't dictate program structure. It potentially overemphasises data over algorithms. Additionally, classes and their relations put limitations and rigidity on reusability and modularity. It's often difficult to reform the code for parallelisation and so on

People also say that Java is stupidly verbose (wordy) and this doesn't need to be the case.

Java is kind of like C, except that lots of things are automated like:

- Exception handling
- Garbage collection

- Default values

It also doesn't use pointers which is really nice.

2.1 Static elements

Static methods can access static data and can change the value of it. Static methods are unable to use non-static data members or call non-static methods directly. Static code blocks can be used to initialise the static data members, and more importantly, they are executed before the main method at the time of class loading.

3 An object's life

Attributes capture what objects can be. Each object has its own copy of attributes and they can be a few things:

- Plain data types (such as bool, char, int, ...)
- References to other objects
- ...

Methods, on the other hand, capture what objects actually do. Methods take parameters and return values. They are able to be *overloaded* (same name, different parameters).

3.1 References