Constants in Java

As you already know, it's good practice to use named constants instead of magic numbers/values in your code. In Python, we used a naming convention (`ALL_CAPS`) to remind ourselves of our intention not to change the value of a constant, even though Python did not enforce this intention. In Java, we can create constants that cannot be reassigned.

How to define constants

A constant is defined like this:

```
public static final <type> <NAME> = <value>;
```

Let's say you wanted to create a `Ball` class and wanted to have two constants, `PI` and `DEFAULT_RADIUS`, in addition to the instance variables `radius` and `color`. Note that each `Ball` object should have its own `radius` and `color`, but `PI` and `DEFAULT_RADIUS` have the same values for all `Ball` objects.

Here's how you might write this class:

```
public class Ball
{
    public static final double PI = 3.14159;
    public static final int DEFAULT_RADIUS = 2;
    private int radius;
    private String color;

    public Ball(int radius, String color)
    {
        this.radius = radius;
        this.color = color;
    }

    public Ball(String color)
    {
        this(DEFAULT_RADIUS, color);
    }

    public double getArea()
    {
        return radius * radius * PI;
    }

    ...
```

What does `final` mean?

The `final` keyword in a variable declaration keeps the variable from being reassigned. So, in the above `Ball` class, we could not have `PI = 3.0` anywhere else in the code.

What does `static` mean?

The `static` keyword is used to avoid making unnecessary copies of the constant. Without the `static` keyword, a variable declared in a class is an *instance variable* or property – i.e. each object from the class has their own independent copy. So every `Ball` has its own `radius` and `color` that is independent of the `radius` and `color` of any other `Ball` object.

However, with the `static` keyword, we have only one copy, and it is *shared* between all objects in the class. So, there's only one `DEFAULT_RADIUS`, and it is shared among all objects of type `Ball`.

How to access constants

Accessing a constant inside the class in which it is declared and defined is easy: just use the constant name. You can see this in the example above, where we accessed the `PI` constant in `getArea()`.

To access a constant in a class other than the one in which it is defined, we follow this pattern `<Class>.<NAME>`. For example, in a class other than `Ball`, if you wanted to access the `DEFAULT_RADIUS` constant defined in `Ball`, you'd access it with `Ball.DEFAULT_RADIUS`.

Note that you don't need to give the name of a `Ball` object to do this; you use the name of the *class* instead. You **can** use an object if you want:

```
Ball theBall = new Ball(3, "Red");
int x = theBall.DEFAULT_RADIUS;
```

but Java programmers are much more likely to just use `Ball.DEFAULT_RADIUS`.

`public` or `private`?

In the examples above, I've made the constants `public` so that they can be accessed outside the class in which they are defined. Because they are constants, this in generally safe. However, you can and should make a constant `private` if you don't want or need anyone outside your class to be able to access the constant.

A warning

One thing to be aware of is that the `final` keyword only keeps the variable from being reassigned. It **does not** make it so that the value remains constant.

Consider the following:

```
import java.util.ArrayList;
import java.util.Arrays;

public class Deck
{
    public static final ArrayList<String> SUITS
        = new ArrayList<String>(Arrays.asList("Clubs", "Diamonds", "Hearts", "Spades"));

    public Deck()
    {
        for (String suit : SUITS) {
            // do something
        }
    }
    ...
```

`SUITS` is an `ArrayList` of `String` objects. Because it is marked `final`, we aren't allowed to reassign `SUITS` like this: `SUITS = new ArrayList<String>()`. However, because `ArrayList`s are mutable, we **can** change the `SUITS` list like this to change the "Hearts" suit to "Loves":

```
SUITS.set(2, "Loves");
```