The class’ own variables

Variables belonging to the class rather than a particular instance

Some variables are not suitable for objects

There are cases when you don’t need or want variables to exist in every separate instance of a class.

If we look at the Member class, obviously name and email are variables which every instance should have their own copy of, so that we can have two instances with different name and email in our system.

But what about stuff that would always be the same for every member ever created? We could use a separator character for the toString method as an example. If the String representation for each object should use “;” between name and email, why would every object need to store this value?

Variables that belong to the class itself

We can use the modifier `static` to declare a variable which belongs to the class independently of any instances which might have been created.

This means that a static variable can be accessed (if they are public) even before any instances have been created, since they belong to the class itself.

The static keyword is the only thing that differs from a declaration of an instance variable:

```
private String name; // instance variable, every object will have its own
copy
private static String separator = ";"; // belongs to the class
```

Bank account example

Instances of a class for a bank account should of course have their own instance variables for balance (so that every bank account object have different balance!).

But the interest rate for one type of bank account could typically be made a static variable - all savings account instances should have the same interest rate and that variable would then be a good candidate for being declared static.

All instances share the static variable interestRate, and when that variable changes (if it does) it would affect all instances at once, which is what we want.

Constants - class variables which don’t change

One use of static variables is to declare a public constant name, which never changes, but allows us to use a descriptive name instead of a value.

A class with mathematical stuff could for instance declare the following:

```
public static final double PI = 3.141592653589793;
```

This would allow the programmers to use PI instead of remembering and using that rather lengthy value.

A variable declared final cannot be re-assigned a new value after intialization.

Accessing a public static variable from other class

The name of a public static variable becomes
<full-class-name>.<variable-name>

The full class name is the package name and class name:

org.progund.constants.Math.PI

package: org.progund.constants

class: Math

variable name: PI

Public static constants have UPPER_CASE_NAME

The convention for naming a public static variable is to use all upper case letters with words separated by underscore:

MAX_VOLUME

FILE_SEPARATOR

etc...

If we use constants like this, and we discover that we need to change the value for e.g. MAX_VOLUME, how many places do we have to change?

Hint: one. (By the way, the only way to change a constant is to recompile)