So far we talked about

* classes with only static features (Utility classes)
* classes with only non-static features

Classes with static and non-static features

* Static features of a class correspond to the CLASS, not to any particular object
* Non-static features correspond only to a particular object

Some applications of combining static and non-static features:

* Keeping track of the number of objects that have been created
* Keep a count of the number of times a method has been called
* Controlling the number of instances of a class that can be created (e.g. a singleton – a kind of class for which only one object can be created)

In addition, we can continue to use static features to do all the same things we used them for in utility classes e.g.

* providing a toolbox
* providing constants

For review:

* static features are part of the class, not part of an object – therefore, they exist even if no instances of a class have been created
* Therefore, they are best for tools/data that must exist even before any objects have been created

Static fields: Best practice is to set an initial value at declaration

e.g.

private static int count = 0;

remember, there is no constructor to set values for static fields!

Counting the number of objects created: If you are using this() to call another constructor, make sure you are only incrementing the instance counter ONCE. On the other hand, if you are not calling this(), make sure EVERY constructor increments the instance counter.

You CANNOT access non-static features within a static method, UNLESS you specify which object you are working on. i.e., “this” is not valid in a static method