# Statics, finals and this

* When you instantiate a class, every object gets unique instances of the properties. This is the normal situation.
* If you don’t need more than one instance of some property, make it static.
* Static properties and methods exist right after you have started your program. Therefore, you can use them *before* you make an instance of the class. You, though, cannot access non-static members from static blocks. The keyword *this* does of course not work in static methods.
* Static properties and methods are *shared* among all objects.
* Static methods should be referenced through the name of the class.
* You should avoid using statics because they can make the program non object-oriented.
* So, when to use?
* Constants are defined with the keyword *final*. When you define a variable to be constant it means that you cannot change the value of the variable.
* Java has two kind of finals. 1) Defined and set anywhere in the program effecting the *block* where it is defined. 2) Defined as a class-member (property) and set in the *constructor*.
* Usually you would like a constant to be constant among all the objects instantiated during the program run. If so, make them static because you only need one instance of the constant.
* If you would like the constant to have different values depending on the instantiation, you cannot make it static.
* **Note:** If you make an object final, you can not change its address (assigning) but you can change the information in it.
* The keyword *this* is used to three things:
  1. Reference the object we are handling.
  2. Calling another constructor.
  3. Referencing the outer class form an inner class (Outer.this.member).
* Examples: Classes\StaticExample, Classes\Statics, Classes\Finals.