Java Notes

Local/Instance/Class Variables

There are three kinds of Java variables:

1. **Local variables** are declared in a method, constructor, or block. When a method is entered, an area is pushed onto the *call stack*. This area contains slots for each local variable and parameter. When the method is called, the parameter slots are initialized to the parameter values. When the method exits, this area is popped off the stack and the memory becomes available for the next called method. Parameters are essentially local variables which are initialized from the actual parameters. Local variables are not visible outside the method.
2. **Instance variables** are declared in a class, but outside a method. They are also called *member* or *field variables*. When an object is allocated in the *heap*, there is a slot in it for each instance variable value. Therefore an instance variable is created when an object is created and destroyed when the object is destroyed. Visible in all methods and constructors of the defining class, should generally be declared private, but may be given greater visibility.
3. **Class / static variables** are declared with the `static` keyword in a class, but outside a method. There is only one copy per class, regardless of how many objects are created from it. They are stored in static memory. It is rare to use static variables other than declared `final` and used as either public or private constants.

characteristic	***Local variable***	***Instance variable***	***Class variable***
Where declared	In a method, constructor, or block.	In a class, but outside a method. Typically `private`.	In a class, but outside a method. Must be declared `static`. Typically also `final`.
Use	Local variables hold values used in computations in a method.	Instance variables hold values that must be referenced by more than one method (for example, components that hold values like text fields, variables that control drawing, etc), or that are essential parts of an object's state that must exist from one method invocation to another.	Class variables are mostly used for *constants*, variables that never change from their initial value.
Lifetime	**Created** when method or constructor is entered. **Destroyed** on exit.	**Created** when instance of class is created with `new`. **Destroyed** when there are no more references to enclosing object (made available for garbage collection).	**Created** when the program starts. **Destroyed** when the program stops.
Scope/Visibility	Local variables (including formal parameters) are visible only in the method, constructor, or block in which they are declared. Access modifiers (`private`, `public`, ...) can *not* be used with local variables. All local variables are effectively private to the block in which they are	Instance (field) variables can been seen by all methods in the class. Which other classes can see them is determined by their declared access: **`private`** should be your default choice in declaring them. No other class can see private instance variables. This is regarded as the best choice.	Same as instance variable, but are often declared `public` to make constants available to users of the class.

	declared. No part of the program outside of the method / block can see them. A special case is that local variables declared in the initializer part of a `for` statement have a scope of the `for` statement.	Define *getter* and *setter* methods if the value has to be gotten or set from outside so that data consistency can be enforced, and to preserve internal representation flexibility. **Default (also called *package* visibility)** allows a variable to be seen by any class in the same package. `private` is preferable. **`public`**. Can be seen from any class. Generally a bad idea. **`protected`** variables are only visible from any descendant classes. Uncommon, and probably a bad choice.	
Declaration	Declare before use anywhere in a method or block.	Declare anywhere at class level (before or after use).	Declare anywhere at class level with `static`.
Initial value	None. Must be assigned a value before the first use.	Zero for numbers, false for booleans, or null for object references. May be assigned value at declaration or in constructor.	Same as instance variable, and it addition can be assigned value in special *static initializer block*.
Access from outside	Impossible. Local variable names are known only within the method.	Instance variables should be declared `private` to promote information hiding, so should not be accessed from outside a class. However, in the few cases where there are accessed from outside the class, they must be qualified by an object (eg, myPoint.x).	Class variables are qualified with the class name (eg, Color.BLUE). They can also be qualified with an object, but this is a deceptive style.
Name syntax	Standard rules.	Standard rules, but are often prefixed to clarify difference from local variables, eg with `my`, `m`, or `m_` (for member) as in `myLength`, or `this` as in `this.length`.	`static public final` variables (constants) are all uppercase, otherwise normal naming conventions. Alternatively prefix the variable with "c_" (for class) or something similar.