JP Mayorga 2014057102 3rd Summary

A nested class is a class that is within the body of another class, therefore a nested class is inside a top class. An abstract class is incompletely implemented and cannot be instantiated. If it final, it cannot have any subclasses and if it is public it can be called from any package. Newly declared methods can implement or override methods from superclasses or superinterface. A synchronize method locks the object before executing the body and unlocks it after. Constructors are almost same as a method but they are used to initialize a new class instance. There are two types of class declaration (normal and enum). If a normal class contains an abstract method, you will find an error. Only use final class if the its definition is complete and it will not have any subclass. Strictfp modifier will make any float or double expression a FP strict. A class is generic if it declares one or more type of variables these variables are called type parameters, where usually one type variable depends of another.

An inner class is a nested class that is not declared static. Using extends means a direct superclass from the actual class and the same applies when using implements, it will be a direct superinterface from the actual class. Members of a class (inherited from superclass, superinterface or declared in its body), members that are private are not inherited by the subclass. Field declarations are used to set the type of field the variable will be used in; a class inherits all non-private fields. Static field = class variables, there will only be one in the field. Non static = instance variables, where a new instance is created for every instance variable declared. Final fields need to be declared and can be applied to either static or not. If you don't declare the final variable, there will be a compile error. Transient variables are not part of the persistent state of the object. A volatile variable will show the same value for all the threads that see this variable (it cannot be volatile and final at the same time).

If the declarator is for a class variable (static) then the variable initializer is evaluated and performed only once when the class is initialized. If it is for an instance variable (non-static), the initializer is evaluated and performed each time a new instance is created. They are also used for local variables and are performed each time the statement is executed. For class initializers, static and final fields are initialized first, these fields will be constant. Initialization expressions for instance variables may use the simple name of any static variable declared in or inherited by the class, even one whose declaration occurs later in the class body. The declaration of a member needs to be in the class body before it's used.