Notes

* A class represents a set of objects which share the same structure and behaviors.
* The class determines the structure of objects by specifying variables that are contained in each instance of the class, and it determines behavior by providing the instance methods that express the behavior of the objects.
* In object-oriented programming, calling a method is often referred to as sending a message to an object.

Sub-Class

* Sub-classing is used mainly in one situation: There is an existing class that can be adapted with a few changes or additions. This is much more common than designing groups of classes and subclasses from scratch.

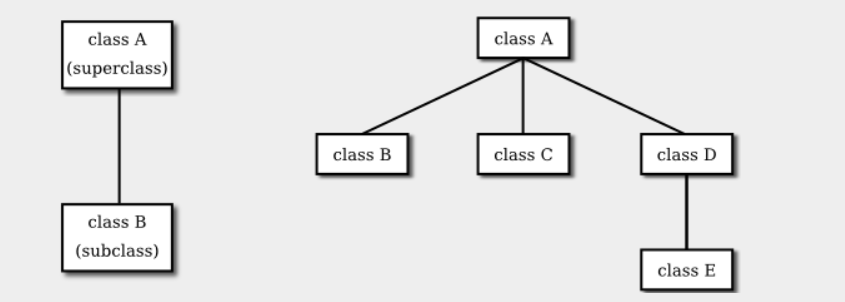
Public class subclass-name extends existing-class-name {

// Changes and additions.

}

Inheritance and Class Hierarchy

* The term inheritance refers to the fact that one class can inherit part or all of its structure and behavior from another class.
* The class that does the inheriting is said to be a subclass of the class from which it inherits. If class B is a subclass of class A, we also say that class A is a superclass of class B. (Sometimes the terms derived class and base class are used instead of subclass and superclass.



* In Java, to create a class named "B" as a subclass of a class named "A", you would write

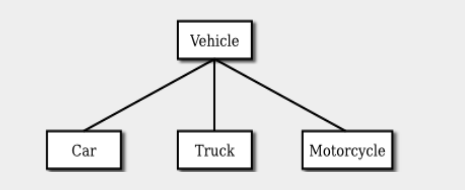
class B extends A {

// additions to, and modifications of,

// stuff inherited from class A

}

* Several classes can be declared as subclasses of the same superclass. The subclasses, which might be referred to as "sibling classes," share some structures and behaviors -- namely, the ones they inherit from their common superclass. The superclass expresses these shared structures and behaviors. In the diagram shown on the right above, classes B, C, and D are sibling classes. Inheritance can also extend over several "generations" of classes. This is shown in the diagram, where class E is a subclass of class D which is itself a subclass of class A. In this case, class E is considered to be a subclass of class A, even though it is not a direct subclass. This whole set of classes forms a small class hierarchy.
* An abstract class is one that is not used to construct objects, but only as a basis for making subclasses. An abstract class exists only to express the common properties of all its subclasses.



* Note, by the way, that the Vehicle class discussed above would probably also be an abstract class. There is no way to own a vehicle as such -- the actual vehicle has to be a car or a truck or a motorcycle, or some other "concrete" type of vehicle.