Java has the property of **inheritance**. A **super class** is a class that contains some method and/ or instance (data) for a **sub-class** to share usage or override.

The keyword “ is a ” shows the parent-child relation of classes.

Super class is more general, sub-class is more specific.

The **substitution principle** states that you can always use a subclass object in place of a superclass object.

Eg: void processVehicle(Vehicle v);

Car myCar = new Car(…);

processVehicle(myCar);

instead of using Vehicle object, Car object can be in place of ().

A subclass inherits all methods that it does not override.

Sub-class header: public class ChoiceQuestion extends Question

The *extends* reserved word indicates that a class inherits from a superclass.

The *private* instance variables of the superclass are inaccessible.

The other keyword: *protect* can all subclass to access.

Use the reserved word *super* to call a superclass method.

super.display();

A subclass can override a superclass method by providing a new implementation. That is, having the same method with the same heading in the class, but content in the {} may be different.

An overriding method can extend or replace the functionality of the superclass method.

A subclass reference can be used when a superclass reference is expected.

But if a reference is in class 1, and the other class contains a method class 1 do not have, then this reference cannot use that method.

In Java, method calls are always determined by the type of the *actual* object, not the type of the variable containing the object *reference*. This is called **dynamic method lookup**.

Dynamic method lookup allows us to treat objects of different classes in a uniform way. This feature is called **polymorphism.**