In a class hierarchy, when a method in a subclass has the same name and type signature as a method in its superclass,

then the method in the subclass is said to override the method in the superclass. When an overridden method is called

from within its subclass, it will always refer to the version of that method defined by the subclass. The version of the

method defined by the superclass will be hidden.

Method overriding occurs only when the names and the type signatures of the two methods are identical.

If they are not, then the two methods are simply overloaded.

(Check display functions in box classes)

Dynamic Method Dispatch:

Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than

compile time. Dynamic method dispatch is important because this is how Java implements run-time polymorphism.

Let's begin by restating an important principle: a superclass reference variable can refer to a subclass object.

When an overridden method is called through a superclass reference, Java determines which version of that method to

execute based upon the type of the object being referred to at the time the call occurs. Thus, this determination is

made at run time.

In other words, it is the type of the object being referred to (not the type of the reference variable)

that determines which version of an overridden method will be executed.

If B extends A then you can override a method in A through B with changing the return type of method to B.