1. **How does inheritance promote the principle of don’t repeat yourself (DRY)?**

Inheritance promotes the principle of “don’t repeat yourself” (DRY) by reducing the need to manually add duplicate methods to similar objects. For example, rather than create a “breath” method for horse, then another “breath” method for chicken, then another for pig, etc, inheritance allows you to create a class called mammal with one “breath” method that horse, chicken, and pig can automatically include in their class (pg 268).

1. **What is the syntax of a derived class that inherits from a base class?**

class DerivedClass : BaseClass (pg 268)

1. **Do all user-defined types (classes and structs) inherit from some base class? If so, what is it?**

All classes inherit from the System.Object class. All structures inherit from an abstract class named System.ValueType. User-defined inheritance applies only to classes, not to structures, and a user-defined class is not allowed to derive from more than one class (pg 269).

1. **What happens if you do not have a default constructor in a base class when creating a derived class?**

If you do not have a default constructor in a base class when creating a derived class, and the base class does not have any non-default constructors, then compiler will simply generate a default constructor (pg 270). However, if the base class does have non-default constructors, you will need to call the correct base-class constructor or else the result will be a compile-time error (pg 271).

1. **Can you assign a variable of a derived class to another variable of its base class? Why or why not?**

Yes, you can assign a variable of a derived class to another variable of its base class because inheritance hierarchy means that you think of the derived variable simply as a special type of base variable; it has everything that the base variable has with a few extra bits (pg 272).

1. **Can you assign a variable of a derived class to another variable of a derived class of its base class? Why or why not?**

It depends: if the other variable is within the same lineage, then it is like the grandchild of the base class and the variables can be assigned. For example, if the class pony is derived from the class horse, then you can safely assign an object of type pony to a variable of type mammal.

However, if the other variable is not of the same lineage, then you cannot assign variables because they were not inherited. For example, even though horse and chicken are both derived from mammal, you cannot assign a variable from horse to a variable from chicken.

1. **Can you assign a variable of a base class to another variable of a derived class? Why or why not?**

You can usually not assign variables of a base class to another variable of a derived class, unless you use the ‘is’ or ‘as’ or ‘cast’ operator to first check whether the base class is of the same type as the derived class. For example:

Horse myHorse = new Horse();  
Mammal myMammal = myHorse;  
Horse myHorseAgain = myMammal as Horse; // OK – myMammal is a horse (pg 272)

1. **Under what circumstances would you want to use the new keyword when defining a method in a derived class?**

When hiding another method of the same name. If a method in a derived class has the same signature as a method in the base class, then the method in the derived class hides the method in the base class so that the latter does not execute. The new keyword turns the warning off (pg 274).

1. **What is a virtual method? Why would you want to define a virtual method?**

A method that is intended to be overridden is called a virtual method. You would want to define a virtual method if you want to override a method in order to provide different implementations of the same method (pg 275).

1. **What does override do? Why does it do it?**

If a base class declares that a method is virtual, a derived class can use the override keyword to declare another implementation of that method. You do it when the methods are intended to perform the same task, but in a class-specific manner (pg 275).

1. **How do you define an extension type?**

You define an extension type by using the ‘this’ keyword. For example:

static class Util  
{  
 public static int Negate(this int i)  
 {  
 return -1;  
 }  
}

**See Answer 12 on Next Page…**

1. **Why do you define an extension type?**

You might want to define an extension type if:

1. You want to use the extension on existing code without having to manually change the definition of every variable you want changed.
2. The type you want to extend is a struct so you cannot use inheritance