**C# Homework 12**

**Question 1**

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

**Answer**

It is a principle of software development aimed to reducing repetition of software patterns, replacing it with abstractions or using data normalization to avoid redundancy. A tool to avoid repetition.

**Question 2**

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

**Answer**

A derived class can access all the non-private members of its base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class. Constructor, destructors and copy constructors of the base class.

**Question 3**

Do all user defined types (classes and structs) inherit from some case class? If so, what is it?

**Answer**

Yes, element of the base class is object

**Question 4**

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

**Answer**

If a base class does not offer a default constructor, the derived class must make an explicit call to a base constructor by using base.

**Question 5**

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

**Answer**

Yes

**Question 6**

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

No. You have a base class of mammal a horse object named big ed

Class Mammal

Class Horse: Mammal

Horse MrEd = new Horse();

Mammal Steven = new Mammal();

//can you do this?

Steven = MrEd;

//answer yes

//because both MrEd and Steven are both Mammals

Horse Reckless = new Horse();

//can you do this?

Reckless = MrEd;

//answer/ yes because they are both horses

What Willie = new Whale();

//can you do this?

Steven = Willie;

//answer/yes

//because they are both mammals

**Question 7**

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

**Answer**

//can you do this?

Willie = Reckless;

//answer/ no

//because/ on is a Horse and one is a Whale

//can you do this?

Mammal TheDonald = new Horse();

//answer/yes

//because a horse is a Mammal

//optional, advanced

Interface LivingBreathingThing;

LivingBreathingThing UncleJoe = new Whale();

**Question 8**

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

**Answer**

When you want to override the warning method.????????????????????????????????????????

**Question 9**

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

**Answer**

A virtual method is one that is declared a virtual in the base class. A method is declared as virtual by specifying the keyword ‘virtual’ in the method signature. A virtual method may or may not have a return type. Virtual methods allow subclasses of the type to override the method.

**Question 10**

What does override do? Why does it do it?

**Answer**

The override keyword is used to extend or modify a virtual/abstract method, property, indexer, or even of case class into derived class. The new keyword is used to hide a method, property, indexer, or event of base class into derived class.

**Question 11**

How do you define an extension type?

**Answer**

An extension method must be defined in a top-level static class. An extension method with the same name and signature as an instance method will not be called. Extension methods cannot be used to override existing methods. The concept of extension methods cannot be applied to fields, properties or events.

Public static RetType MethodName(this int Raquel)

Raquel - parametername

Int =The name of the class you are extending

This - is a keyword that signifies an extension method

**Question 12**

Why do you define an extension type?

**Answer**

Extension methods enable you to ‘add’ methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type.

**Question 13**

(Not in book.) Explain the Liskov substitution princliple.

**Answer**

Substitutability is a principle in object-oriented programming stating that, in computer program, if S is a subtype of T, then objects of type T may be replaced with objects of type S without altering any type of the desirable properties of the program.