**C# Homework 13**

**Question 1**

What is an interface as the term is used on object-oriented programming?

**Answer**

By using an interface, you can completely separate the names and signatures of the methods of a class from the method’s implementation. An interface is a description of all functions that an object must have in order to be an ‘X’. The purpose of interfaces is to allow the computer to enforce these properties and to know that an object of TYPE T (whatever the interface is) must have functions call X, Y, Z, etc. it separates the what from the how.

**Question 2**

How do you define an interface?

**Answer**

Defining an interface is syntactically similar to defining a class, except that you use the interface keyword instead of the class keyword. Within the interface, you declare methods exactly as in a class or structure, except that you never specify an access modifier.

Int Calculate(int rhs, int lhs);

**Question 3**

Can an interface have variables, fields, or properties?

**Answer**

No, an interface does not have fields; in other words we can’t declares variables in an interface. So we can’t declare a field in the interface. When an interface is inherited by a class or struct then need to implement all members of the interface in that class.

**Question 4**

How do you define a method in an interface?

**Answer**

An abstract method within an interface is followed by a semicolon, but no braces (an abstract method does not contain an implementation). Default methods are defined with the default modifier, and static methods with the static keyword.

Int Calculate(int rhs, int lhs);

Replace all the implementation with a semicolon

**Question 5**

Can you instantiate an object through an interface? Why or why not?

**Answer**

Interfaces cannot be instantiated, but rather are implemented. A class that implements an interface must implement all of the non-default methods described in the interface or be an abstract class.

Interface IMammal{}

Class Human {}

IMammal Linh = new Human();

Linh is a IMammal!!

Interface IShape {}

Class Square : IShape{ Area( sideLength) {sideLength \* sideLength}}

Classs Circle : IShape{ circumference(int radius) {radius \* radius \* Math.PI}}

IShape s = new Square();

IShape c = new Circle();

//s can call Draw() but not Area() because it is actually an IShape

//c can call Draw() but not circumference() because it is actually an IShape

**Question 6**

Using the new keyword, can you declare a reference to an interface?

**Answer**

Yes, you can. Look at previous example on #5.

**Question 7**

Can an object inherit from multiple interface? Can a class implement multiple interfaces? If so, ho can it do so?

**Answer**

Yes, an interface contains variables and methods like a class but the methods in an interface are abstract by default unlike a class. Multiple inheritance by interface occurs if a class implements multiple interfaces or also if an interface itself extend multiple interfaces.

**Question 8**

What does it mean to explicitly implement an interface?

**Answer**

Explicit implementation is also used to resolve cases where two interfaces each declare different members of the same name such as a property and a method. To implement both interfaces, a class has to use explicit implementation either for the property P, or the method P, or both, to avoid a compiler error.

**Question 9**

What are the restrictions on interfaces?

**Answer**

The members of an interface must be methods, properties, events, or indexers. An interface cannot contain constants, fields, operators, instance constructors, destructors, or types nor can an interface contain static members of any kind. All interface members implicitly have public access.

**Question 10**

What is the difference between an abstract class and an interface?

**Answer**

An abstract class allows you to create functionality that subclasses can implement or override. An interface only allows you to define functionality, not implement it. And whereas a class can extend only one abstract class, it can take advantage of multiple interfaces.

**Question 11**

What is an abstract method?

**Answer**

An abstract method is a method without a body. The implementation of an abstract method is done by a derived class. When the derived class inherits the abstract method from the abstract class, it must override the abstract method. The abstract method is declared by adding the abstract modifier the method.

**Question 12**

What is a sealed class?

**Answer**

Sealed classes are used to restrict the inheritance feature of object oriented programming. Once class is defined as a sealed class, this class cannot be inherited. The sealed modifier is used to declare a class as sealed. If a class is derived from a sealed class, compiler throws an error.

**Question 13**

What is a sealed method?

**Answer**

Prevent overriding a method of class. This is a method that is declared with the keyword sealed and is always used with combination of override keyword. Derived classes will not be able to override this method as it is sealed for overriding.