1. what are the six combinations of access modifier keywords and what do they do?

Ans:

Public: Objects that implement public access modifiers are accessible from everywhere in a project without any restrictions.

Private: Objects that implement private access modifier are accessible only inside a class or a structure. As a result, we can’t access them outside the class they are created

Protected: The protected keyword implies that the object is accessible inside the class and in all classes that derive from that class.

Internal: For Internal keyword, the access is limited exclusively to classes defined within the current project assembly.

Protected Internal: The protected internal access modifier is a combination of protected and internal.

Private Protected: The private protected access modifier is a combination of the private and protected keywords.

1. What is the difference between the static, const, and read-only keywords when applied to a type of member?

Ans:

The static keyword is used to make members static that can be shared by all the class objects.

Constant and read-only keyword is used to make a field constant which value cannot be modified.

1. What does a constructor do?

Ans:

A constructor is a special method of a class or structure in object-oriented programming that initializes a newly created object of that type.

1. Why is the partial keyword useful?

Ans:

The partial keyword indicates that other parts of the class, struct, or interface can be defined in the namespace. All the parts must use the partial keyword. All the parts must be available at compile time to form the final type.

1. What is a tuple?

Ans:

Tuples are used to store multiple items in a single variable.

1. What does the C# record keyword do?

Ans:

Record Keyword is used to define a reference type that provides built-in functionality for encapsulating data.

1. What does Overloading and Overriding mean?

Ans:

Method Overloading is a Compile time polymorphism. In method overloading, more than one method shares the same method name with a different signature in the class.

Method Overriding is a Run time polymorphism. In method overriding, the derived class provides the specific implementation of the method that is already provided by the base class or parent class.

1. What is the difference between a field and a property?

Ans:

A field is a variable of any type that is declared directly in a class.

A property is a member that provides a flexible mechanism to read, write or compute the value of a private field.

1. How do you make a method parameter optional?

Ans:

* Use parameter arrays
* Default parameter
* Use optional attribute
* Method Overriding

1. What is an interface and how is it different from abstract class?

Ans:

Interfaces are a kind of code contract, which must be implemented by a concrete class.

Abstract classes are similar to normal classes, with the difference that they can include abstract methods, which are methods without a body. Abstract classes cannot be instantiated.

1. What accessibility level are members of an interface?

Ans:

Private.

1. True/False. Polymorphism allows derived classes to provide different implementations of the same method.

Ans:

True.

1. True/False. The override keyword is used to indicate that a method in a derived class is providing its own implementation of a method.

Ans:

True.

1. True/False. The new keyword is used to indicate that a method in a derived class is providing its own implementation of a method.

Ans:

False.

1. True/False. Abstract methods can be used in a normal (non-abstract) class.

Ans:

False.

1. True/False. Normal (non-abstract) methods can be used in an abstract class.

Ans:

True.

1. True/False. Derived classes can override methods that were virtual in the base class.

Ans:

True.

1. True/False. Derived classes can override methods that were abstract in the base class.

Ans:

False.

1. True/False. In a derived class, you can override a method that was neither virtual non abstract in the base class.

Ans:

False.

1. True/False. A class that implements an interface does not have to provide an implementation for all of the members of the interface.

Ans:

True.

1. True/False. A class that implements an interface is allowed to have other members that aren’t defined in the interface.

Ans:

True.

1. True/False. A class can have more than one base class.

Ans:

True.

1. True/False. A class can implement more than one interface.

Ans:

True.