Object Oriented Programming - Test Your Knowledge

1. What are the six combinations of access modifier keywords and what do they do?
Public, protected internal, protected, internal, private protected, private
Public: All classes can access the public.
protected internal: can be accessed within the class, derived class, non-derived(same assembly), derived class(different assembly)
Protected: can be accessed within the class, derived class(same assembly), and derived class(different assembly)
Internal: cannot be accessed by different assemblies for both derived and non-derived classes. It can be accessed within the same class and other classes within the same assembly.
Private protected: can only be accessed within the class and derived class(same assembly)
Private: can only be accessed within the class.

2.What is the difference between the static, const, and readonly keywords when applied to a type member?
Static is used to define a member that belongs to the type itself. It is not an instance of the type. For example, Math.PI is a static member of the class Math.
Const is used to define a constant value that cannot be changed once initialized.
Readonly is also used to define a value that cannot be changed once initialized. However, a Readonly variable can be initialized during runtime. It can be static or instance-level.

3. What does a constructor do?

A constructor is special member function shares the name of the class

Constructor is used to create instance of the class

Constructor can be overloaded with multiple parameters

If there is no constructor then compiler provides a default constructor

A default constructor is always replaced by the custom customer

Constructor is used to initialize the class fields

Constructor can not have any return type not even void

4. Why is the partial keyword useful?
The partial keyword is used to split a class, struct, or interface definition into multiple parts, each in a separate file. It is useful because it improves organization, readability, collaboration, and helps with separating auto generated code from written code.

5. What is a tuple?
Tuple is similar to an array or a list, but it allows elements of different data types. For example, var tuple = {3, “string”, 5.21)

6. What does the C# record keyword do?
A record is similar to a class, but it defines immutable types. within a record, you only need to define the properties. Once the properties are defined, they cannot be changed. In addition, two records with the same value of properties are considered equal. And an instance of a record always has a formatted output.

7. What does overloading and overriding mean?
Overloading refers to in a class, there are methods that have the same name but with different parameters. It is a type of compile time polymorphism.

Overriding refers to methods in the base class and subclass that share the same name and same parameters. The methods in the subclass overrides the methods in the base class. It is a type of run time polymorphism.

8. What is the difference between a field and a property?
A **property** is a public data member that reads and writes private fields. It could be a getter which gets value from the field or a setter that sets the value of a field.

A **field** is a member that you first define in a class. If it is a private field, we will use getter and setter to access its value.

9. How do you make a method parameter optional?
you can make a method parameter optional by providing a default value for the parameter.

10. What is an interface and how is it different from abstract class?
Abstract classes can have *constants, members, method stubs (methods without a body) and defined methods*, whereas interfaces can only have *constants* and *methods stubs*.

11. What accessibility level are members of an interface?

the members of an interface are always implicitly public and cannot have any access modifiers. When a class implements an interface, it must provide an implementation for all the members of the interface, and these members are also implicitly public.

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

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

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

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

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

17. True/False.Derived classes can override methods that were virtual in the base class.
18. True/False.Derived classes can override methods that were abstract in the base class.
19. True/False.In a derived class, you can override a method that was neither virtual non abstract in the base class.
20. True/False. A class that implements an interface does not have to provide an implementation for all of the members of the interface.

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

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

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