**03 Object-Oriented Programming**

**Test your knowledge**

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

**Public:** The type or member can be accessed by any other code in the same assembly or another assembly that references it.

**Private:** The type or member can be accessed only by code in the same **class** or **struct**.

**Protected:** The type or member can be accessed only by code in the same class, or in a class that is derived from that class.

**Internal:** The type or member can be accessed by any code in the same assembly, but not from another assembly. In other words, internal types or members can be accessed from code that is part of the same compilation.

**Protected Internal:** The type or member can be accessed by any code in the assembly in which it's declared, or from within a derived class in another assembly.

**Private Protected:** The type or member can be accessed by types derived from the class that are declared within its containing assembly.

1. What is the difference between the **static, const**, and **readonly** keywords when applied to a type member?

**Const** refers to a variable that can only be initialized during compile time and then becomes immutable (cannot change its value)

**Readonly** refers to a variable that can only be initialized either during compile time or runtime, but then cannot have its values changed.

**Static** refers to a variable, method or class belongs to the type itself rather than a specific object

1. What does a constructor do?

A constructor initializes an instance of a class and the properties of that instance.

1. Why is the partial keyword useful?

The partial keyword indicates that other parts of the class, struct, or interface can be defined in the namespace.

1. What is a tuple?

A tuple can represent a database record, and its components can represent individual fields of the record.

1. What does the C# record keyword do?

A new reference type other than structs or classes.

1. What does overloading and overriding mean?

Overloading is when the same class has the same name for two methods that may take different number/order of/types of parameters.

Overrriding is when a child class has the same name for a method as its parent class.

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

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?

Default parameter values

Named Parameter

Use OptionalAttribute

Method Overloading

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

Interfaces supports multiple inhertance but abstract classes cannot

Interfaces can not have instanced Constructors but abstract classes can (theoretically)

Interfaces has by default all members to be considered "abstract" and public, but abstracted classes.

Interfaces can not have fields but abstract classes can have fields

1. What accessibility level are members of an interface?

public

1. **True**/~~False~~. Polymorphism allows derived classes to provide different implementations of the same method.
2. **True**~~/False~~. The override keyword is used to indicate that a method in a derived class is providing its own implementation of a method.
3. ~~True/~~**False**. The new keyword is used to indicate that a method in a derived class is providing its own implementation of a method.
4. True/**False**. Abstract methods can be used in a normal (non-abstract) class.
5. **True**/False. Normal (non-abstract) methods can be used in an abstract class.
6. **True**/False. Derived classes can override methods that were virtual in the base class.
7. True/**False**. Derived classes can override methods that were abstract in the base class.
8. True/**False**. In a derived class, you can override a method that was neither virtual non abstract in the base class.
9. True/**False**. A class that implements an interface does not have to provide an implementation for all of the members of the interface.
10. **True**/False. A class that implements an interface is allowed to have other members that aren’t defined in the interface.
11. True/**False**. A class can have more than one base class.
12. **True**/False. A class can implement more than one interface.