Polymorphism Interview Questions and Answers in C#

In this article, I am going to discuss the most frequently asked **Polymorphism Interview Questions and Answers in C#**. Please read our previous article where we discussed the most frequently asked **Abstract and Sealed Class Interview Questions in C#** with Answers. As part of this article, we are going to discuss the following Polymorphism Interview Questions in C# with Answers.

1. **What is Polymorphism in C#?**
2. **Explain the different types of Polymorphism in C#?**
3. **What is compile-time Polymorphism in C#?**
4. **What is Runtime Polymorphism in C#?**
5. **Explain different types of Overloading in C#?**
6. **What is function overloading?**
7. **When should we overload methods in C#?**
8. **What are the advantages of using overloading OR what are the disadvantages if we define methods with a different name?**
9. **When is a method considered as an overloaded method?**
10. **Can we overload methods in the same class?**
11. **What is the execution control flow of overloaded methods?**
12. **What is inheritance based overloading?**
13. **What is the function/method overriding?**
14. **When must a method be overridden?**
15. **When is a sub-class method treated as an overriding method?**
16. **How can we override a parent class method under child class?**
17. **How can we execute the superclass method if it is overridden in the sub-class?**
18. **What is the difference between function overloading and function overriding?**
19. **What is method hiding?**
20. **What is the difference between Method Overriding and Method Hiding?**
21. **When can a derived class override a base class member?**
22. **What is the difference between a virtual method and an abstract method?**
23. **What is the difference between a virtual method and an abstract method?**

What is Polymorphism in C#?

Polymorphism is one of the primary pillars of object-oriented programming. It allows us to invoke derived class methods through a base class reference variable during runtime.

In the base class, the method is declared as virtual, and in the derived class, we override the same method. The virtual keyword indicates that the method can be overridden in any derived class.

The word Polymorphism is derived from the Greek word, where Poly means many, and morph means faces/ behaviors. So polymorphism means the ability to take more than one form.

The same function/ operator will show different behaviors when passed different types of values or the different number of values. So in simple words, we can say that behaving in different ways depending upon the input received is known as polymorphism i.e. whenever the input changes automatically the output or the behavior also changes.

We can implement polymorphism in our application using three different approaches like

1. Overloading
2. Overriding
3. Hiding

Overloading again is of three types

1. **Method overloading**
2. Operator overloading
3. Constructor overloading

Explain the different types of Polymorphism in C#?

There are two types of polymorphism

1. Static polymorphism/compile-time polymorphism /early binding
2. Dynamic polymorphism / Run-time polymorphism /late binding

What is compile-time Polymorphism in C#?

This is one of the frequently asked **C# Polymorphism interview** questions. In the case of compile-time polymorphism, the object of the class recognizes which method to be executed for a particular method call at the time of program compilation and binds the method call with method definition.

This happens in the case of overloading because in the case of overloading each method will have a different signature and basing on the method call we can easily recognize the method which matches the method signature. It is also called static **polymorphism** or early binding. Static polymorphism is achieved by using function overloading and operator overloading

What is Runtime Polymorphism in C#?

This is also one of the frequently asked Polymorphism interview questions in C#. In the case of runtime **polymorphism** for a given method call, we can recognize which method has to be executed exactly at runtime but not in compilation time because in the case of overriding and hiding we have multiple methods with the same signature. So which method to be given preference and executed that is identified at runtime and binds the method call with its suitable method. It is also called dynamic polymorphism or late binding. Dynamic polymorphism is achieved by using function overriding.

Explain different types of Overloading in C#?

Again overloading is classified into three types, such as

1. **Method overloading / Function overloading**
2. Constructor overloading
3. Operator overloading.

What is function overloading in C#?

Function overloading and **method overloading** terms are used interchangeably. **Method overloading** allows a class to have multiple methods with the same name but with a different signature. So in C# functions can be overloaded based on the number, type (int, float, etc), and kind (Value, Ref or Out) of parameters.

The signature of a method consists of the name of the method and the type, kind (value, reference, or output), and the number of its formal parameters. The signature of a method does not include the return type and the params modifiers. So it is not possible to overload a method just based on the return type and params modifier.

A function overloading can be compared with person overloading. If a person has already some work to do and if we are assigning additional work to the person then the person will be overloaded.

In the same way, a function will have already some work to do and if we assign different work to the same function, then we say the function is overloaded. It is an approach of

defining multiple methods with the same method name by changing the signature. Changing the signature means we can either change the no of parameters being passed to the method or type of parameters being passed to the method or order of parameters being passed to the function.

When should we overload methods?

To execute the same logic with different types of arguments we should overload methods. For example to add two integers, two floats and two strings we should define three methods with the same name as shown in the below application

What are the advantages of using overloading OR what are the disadvantages if we define methods with a different name?

If we overload the method, the user of our application gets comfort feeling in using the method with the impression that he/she calling one method bypassing different types of values.

The best example for us is the “WriteLine()” method. It is an overloaded method, not a single method of taking different types of values.

When is a method considered as an overloaded method?

If two methods have the same method name those methods are considered overloaded methods.

Then the rule we should check is both methods must have different parameter types/list/order. But there is no rule on return type, non-accessibility modifier and accessibility modifier means overloading methods can have their own return type, non-accessibility modifier, and accessibility modifier because overloading methods are different methods

Can we overload methods in the same class?

Yes, it is possible no CE, no RE. Methods can be overloaded in the same or in super and subclasses because overloaded methods are different methods.

But we can’t override the method in the same class it leads to CE: “method is already defined” because overriding methods are the same methods with a different implementation.

What is the execution control flow of overloaded methods?

The compiler always checks for the called method definition in reference variable type class with the given argument type parameter. So in searching and executing a method definition, we must consider both reference variable type and argument type. Referenced variable type for deciding from which class method should be to bind. Argument type for deciding which overloaded method should be a bind. For example:

```
B b = new B();
A a = new B();
```

b.m1(50) => b.m1(int); In this method call we should search m1() method definition in B class with integer parameter at the time of program compilation and bind that method definition.

a.m1(50); => a.m1(int); In this method call we should search m1() method defined in class A with int parameter not in class B even though the object is B.

What is inheritance based overloading?

A method that is defined in a class can be overloaded under its child class if we overload a method in this process we call it inheritance-based overloading.

What is the function/method overriding?

Redefining the superclass non-static method in the subclass with the same prototype is called **method overriding**. The overriding method is always executed from the class of the current object.

In object-oriented programming **method overriding** is a language feature that allows a subclass to provide a specific implementation of a method that is already provided by one of its superclasses.
The implementation of the subclass overrides (replaces) the implementation of superclass methods. So the overridden method is always executed from the object whose object is stored in the reference variable. The superclass method is called the overridden method and the sub-class method is called the overriding method.

When must a method be overridden?

If superclass method logic is not fulfilling sub-class business requirements, the subclass should override that method with the required business logic. Usually, superclass methods are defined with generic logic which is common for all sub-classes.

When is a sub-class method treated as an overriding method?

If a method in sub-class contains the same signature as the superclass non-private method then the subclass method is treated as the overriding method and the superclass method is treated as the overridden method.

How can we override a parent class method under child class?

If we want to override a parent class method in its child class, first the method in the parent class must be declared as virtual by using the keyword virtual then only the child classes get the permission for overriding that method. Declaring the method as virtual is marking the method is overridable.
If the child class wants to override the parent class virtual method then the child class can do it with the help of the override modifier. But overriding the method under child class is not mandatory for the child classes. The Syntax is given below:

```
Class1:
Public virtual void show() //virtual function (overridable)
Class2: Class1
Public override void show() //overriding
```

Even if the method declared as virtual the child class may or may not override the method In overriding, the parent class defines a method as virtual and gives it to the child class to consume that method. So the child class now consumes the method as it is or overrides that method as per the requirement of the child class. So overriding the parent class virtual method under a child class is only optional.

How can we execute the superclass method if it is overridden in the sub-class?

After re-implementing parent class methods under child class, the object of the child class calls its own methods but not its parent class method, whereas if we want to still consume or call the parent class's methods from child class, it can be done in two different ways.
By creating a parent class object under the child class, we can call the parent class methods from the child class. Or by using the base keyword, we can call parent class methods from child class, but this and base keyword cannot be used under the static block.

What is the difference between function overloading and function overriding?

FUNCTION OVERLOADING	FUNCTION OVERRIDING
It is an approach of defining multiple methods with the same name and different signature.	It is an approach of defining multiple methods with the same name and same signature.
Overloading a method can be performed within a class or within the child classes also.	Overriding of methods is not possible within the same class it must be performed under parent and child classes.
To overload a parent class method under child class, the child class does not require permission from the parent.	To override a parent class method under child class first the child class requires explicit permission from its parent.
This is all about defining multiple behaviors to a method.	This is all about changing the behavior of a method.
Used to implement static polymorphism.	Used to implement dynamic polymorphism.
This is a code refinement technique.	This code replacement technique.
No separate keywords are used to implement function overloading.	Use the virtual keyword for base class function and override keyword in derived class function to implement function overriding.

What is method hiding?

Use the new keyword to hide a base class member. We will get a compile warning if we miss the new keyword. This is also used for re-implementing a parent class method under child class. Reimplementing parent class methods under child classes can be done using two different approaches, such as

1. **Method overriding**
2. **Method hiding**

In the first case, we re-implement the parent class methods under child classes with the permission of the parent class because here in parent class the method is declared as virtual giving the permission to child classes for overriding the methods.
In the 2nd approach, we re-implement the method of parent class even if those methods are not declared as virtual that is without parent permission we are re-implementing the methods. The Syntax is given below.

Class1:
Public void display()
Class2 : Class1
Public new void display()

Using the new keyword for re-implementing the methods in the child class is optional and if used will give information to hiding.

What is the difference between Method Overriding and Method Hiding?

This is one of the frequently asked **Polymorphism** interview questions in C#. A parent class method can be redefined under its child class using two different approaches.

1. **Method Overriding.**
2. **Method Hiding.**

In **Method overriding**, the parent class gives permission for its child class to override the method by declaring it as **virtual**. Now the child class can override the method using the **Override** keyword as it got permission from the parent. The parent class methods can be redefined under child classes even if they were not declared as **Virtual** by using the **'new'** keyword.

In **method overriding** a base class reference variable pointing to a child class object will invoke the overridden method in the child class. In method hiding a base class reference variable pointing to a child class object will invoke the hidden method in the base class.
For hiding the base class method from the derived class simply declare the derived class method with the new keyword. Whereas in C#, for overriding the base class method in a derived class, we need to declare the base class method as virtual and the derived class method as the override.
If a method is simply hidden then the implementation to call is based on the compile-time type of the argument “this”. Whereas if a method is overridden then the implementation to be called is based on the run-time type of the argument “this”. New is reference-type specific, overriding is object-type specific.

When can a derived class override a base class member?

A derived class can override a base class member only if the base class member is declared as virtual or abstract.

What is the difference between a virtual method and an abstract method?

A virtual method must have a body whereas an abstract method should not have a body.

Can fields inside a class be virtual?

No, Fields inside a class cannot be virtual. Only methods, properties, events, and indexers can be virtual.

Can you access a hidden base class method in the derived class?

Yes, Hidden base class methods can be accessed from the derived class by casting the instance of the derived class to an instance of the base class as shown in the example below.

```
public class BaseClass
{
public virtual void Method()
{
Console.WriteLine("I am a base class method.");
}
}
public class DerivedClass : BaseClass
{
public new void Method()
{
Console.WriteLine("I am a child class method.");
}
public static void Main()
{
DerivedClass DC = new DerivedClass();
((BaseClass)DC).Method();
}
}
```

What is the difference between a virtual method and an abstract method?

This is one of the frequently asked C# Polymorphism interview questions. A virtual method must have a body whereas an abstract method should not have a body. A Base class virtual method may or may not be overridden in the Derived class whereas a Base class Abstract method has to be implemented by the derived class.

In the next article, I am going to discuss the most frequently asked **Partial Class and Nested Types Interview Questions in C#** with Answers. Here, in this article, I try to explain the most frequently asked **Polymorphism Interview Questions and Answers** in C#. I hope you enjoy this Polymorphism Interview Questions and Answers in C# article. I would like to have your feedback. Please post your feedback, question, or comments about this article.