1. Что такое производный и базовый классы?

Производный класс это – класс который наследует все те же свойства, методы, поля, которые есть в базовом классе. Единственное, что не передается при наследовании, это конструкторы базового класса (*тип доступа к производному классу должен быть таким же, как и у базового класса, или более строгим*). Производный класс может иметь доступ только к тем членам базового класса, которые определены с модификаторами **public, internal, protected и protected internal.**

Базовый класс это - обычный класс в котором содержатся поля, свойства и методы. В С# не поддерживается множественное наследование, класс может наследоваться только от одного класса. Хотя проблема множественного наследования реализуется с помощью концепции интерфейсов.

2. BASE()

С помощью ключевого слова base можно вызвать конструктор любой формы, определяемой в базовом классе, причем выполняться будет лишь тот конструктор, параметры которого соответствуют переданным аргументам.

2. В чем заключена основная задача наследования?

Производный класс наследует все переменные, методы, свойства и индексаторы, определяемые в базовом классе, добавляя к ним свои собственные элементы. Наследование предохраняет инкапсуляцию, а потому приватные члены никогда не могут быть доступны через ссылку на объект.

3. Пусть базовый класс содержит метод basefunc(), а производный класс не имеет метода с таким именем. Может ли объект производного класса иметь доступ к методу basefunc()? Если да, то при каких условиях?

Да, может, если метод имеет модификатор доступа **public, internal, protected и protected internal.**

Доступ к членам базового класса во вложенном классе можно получить даже если эти базовые члены переопределены в производном классе, при помощи ключевого слова **[base](https://msdn.microsoft.com/ru-ru/library/hfw7t1ce(v=vs.90).aspx)**.

4. Напишите объявление конструктора без аргументов для производного класса B, который будет вызывать конструктор без аргументов базового класса A.

public B : base()

{

//…

//...

}

5. Что такое полиморфизм? Приведите пример.

*1 вариант*

public summ (int a, int b)

{

return a + b;

}

public summ (int a, int b, int c)

{

return a + b - c;

}

*2 вариант*

Class person

{

virtual public Writing (string str)

{

Console.WriteLine (str + “this is overrided method”);

}

}

Class Employee : Person

{

override public Writing (string str)

{

Console.WriteLine (str + “this is overrided method”);

}

}

Полиморфизм – это взаимозаменяемость объектов с одинаковым интерфейсом. Полиморфизм предполагает определение полиморфного интерфейса – набора полей, свойств, методов одного типа, которые могут быть переопределены или реализованы в производных классах.

Методы и свойства которые мы хотим переопределить помечаются ключевым словом **virtual.** Они представляют полиморфный интерфейс. При переопределении в классе- наследниеке используется ключевое слово **override**

6. Определите назначение виртуальных функций.

**Виртуальным** называется такой метод, который объявляется как **virtual** в базовом классе. Виртуальный метод отличается тем, что он может быть переопределен в одном или нескольких производных классах. У каждого производного класса может быть свой вариант виртуального метода. При их вызове по ссылке на базовый класс в C# определяется именно тот вариант виртуального метода, который следует вызывать, исходя из типа объекта, к которому происходит обращение по ссылке. Иными словами, вариант выполняемого виртуального метода выбирается по типу объекта, а не по типу ссылки на этот объект.

7. Кому доступны переменные с модификатором protected?

Любым классам-наследникам.

8. Наследуются ли переменные с модификатором private?

Да, но они не являются доступными.

9. As, is – что это, как применяется?

Если преобразование типов требуется произвести во время выполнения проги, но не генерировать исключение, если же исход этого преобразования окажется неудачным.

*выражение* ***as*** *тип* (неявное приведение)

Конкретный тип объекта можно определить с помощью оператора is.

*выражение* ***is*** *тип* (проверка на принадлежность\на возможность принадлежности)

10. Поддерживает ли C# множественное наследование?

Не поддерживает.

11. Можно ли запретить наследование от класса?

Да с помощью sealed.

12. Можно ли разрешить наследование класса, но запретить перекрытие метода?

Да, можно если в переопределяемом методе писать ***sealed override***.

13. Что такое абстрактный класс?

Абстрактный класс – это такой же класс, как и обычный со своими полями, свойствами и методами. Но при определении методов используется ключевое слова abstract. А также нельзя использовать конструкторы абстрактного класса для создания объекта, следовательно, и объектов абстрактного класса быть не может.

14. В каком случае вы обязаны объявить класс абстрактным?

Если класс имеет хотя бы одно абстрактное свойство или метод он обязан быть определён как **abstract**.

15. В чем разница между абстрактными и виртуальными классами? Между виртуальными и абстрактными методами?

Нету виртуального класса.

Виртуальный метод имеет реализацию и может быть переопределён в классе-наследнике. Абстрактный метод не имеет реализации, только описание и обязан быть реализован в производном классе.

16. Какие компоненты класса могут быть виртуальными?

Методы, свойства, индексаторы, события.

17. Что такое интерфейс?

Интерфейсы позволяют определить некоторый функционал не имеющие конкретной реализации. Затем этот функционал реализуют классы, применяющие этот интерфейс. Интерфейсы так же, как и классы могут содержать свойства методы, события, методы, но без конкретной реализации. Все члены интерфейсов методы и свойства не имею модификаторов доступа, но фактически по умолчанию они имеют доступ **public.**

18. Как работать с объектом через унаследованный интерфейс?

Если есть несколько унаследованных интерфейсов, то при вызове методов интерфейса необходимо явно указать реализацию какого метода мы хотим использовать (например, с помощью операции приведения типов).

19. Приведите пример явной реализации интерфейса.

Public interface IName

{

Void WriteName();

}

Public interface IFamily

{

Void WriteName();

}

Class UserInfo : IName, IFamily

{

Void IName.WriteName ()

{

Console.WriteLine(“Enter your name: ”);

}

Void IFamily.WriteName ()

{

Console.WriteLine(“Enter your surname: ”);

}

20. Почему нельзя указать модификатор видимости для методов интерфейса?

Все методы интерфейса должны быть реализованы в производном классе, для этого они (методы и свойства) все по умолчанию имеют модификатор **public.**

21. Можно ли наследовать от нескольких интерфейсов?

Можно

22. Назовите отличия между интерфейсом и абстрактным классом.

Если классы относятся к единой системе классификации – выбираем абстрактный класс, иначе выбираем интерфейс.

Если нужно чтобы все производные классы на всех уровнях наследования имели общую реализацию – абстрактный класс. Если необходимо определить функционал для группы разрозненных объектов, которые могут быть никак не связаны между собой (+ проектируем небольшой функциональный тип). Интерфейсы поддерживают «множественное наследование», а абстрактные классы нет.

23. Для чего используются стандартные интерфейсы ICloneable, IComparable, IComparer, lEnumerable?