1. **Назовите принципы ООП. Поясните каждый из них.**

Наследование - процесс, благодаря которому один объект  
может наследовать (приобретать)  
свойства от другого объекта.

Инкапсуляция - механизм, связывающий вместе данные и  
код, обрабатывающий эти данные, и  
сохраняющий их от внешнего  
воздействия и ошибочного использования

Абстракция подразумевает разделение и  
независимое рассмотрение интерфейса  
и реализации

Полиморфизм – возможность объектов с одинаковой спецификой иметь различную реализацию.

**2. Назовите класс .NET, от которого наследуются все классы**

System.Object

**3. Охарактеризуйте открытые методы System.Object.**

ToString()

Метод ToString() возвращает символьную строку, содержащую описание того объекта, для которого он вызывается.

GetHashCode()

Этот метод используется, когда объект помещается в структуру данных, известную как карта (map), которая также называется хеш-таблицей или словарем.

Equals() и ReferenceEquals()

По умолчанию метод Equals (object) определяет, ссылается ли вызывающий объект на тот же самый объект, что и объект, указываемый в качестве аргумента этого метода, т.е. он определяет, являются ли обе ссылки одинаковыми.

GetType()

Этот метод возвращает экземпляр класса, унаследованный от System.Type.

**4. Охарактеризуйте закрытые методы System.Object.**

Finalize()

Назначение этого метода в C# примерно соответствует деструкторам С++, и он вызывается при сборке мусора для очистки ресурсов, занятых ссылочным объектом. Реализация Finalize() из Object на самом деле ничего не делает и игнорируется сборщиком мусора. Обычно переопределять Finalize() необходимо, если объект владеет неуправляемыми ресурсами, которые нужно освободить при его уничтожении.

Clone()

Этот метод создает копию объекта и возвращает ссылку на эту копию (а в случае типа значения — ссылку на упаковку)

**5. Приведите пример определения класса.**

class Person

    {

        public string name; // имя

        public int age;     // возраст

        public void GetInfo()

        {

            Console.WriteLine($"Имя: {name}  Возраст: {age}");

        }

**6. Какие ключевые слова можно использовать при определении класса?**

Атрибуты и спецификаторы (public, private, static).

**7. В чем отличие между объектом и классом?**

Класс – это некоторое абстрактное понятие - шаблон, по которому определяется форма объекта  
Объект – это физическая реализация класса(шаблона).

**8. Что такое конструктор? Когда вызывается конструктор?**

Конструкторы — это специальные методы, позволяющие корректно инициализировать новый экземпляр типа. Создание экземпляра объекта ссылочного типа  
1) выделяется память для полей данных экземпляра  
2) инициализируются служебные поля  
3) вызывается конструктор экземпляра, устанавливающий исходное состояние нового объекта. Память всегда обнуляется до вызова конструктор экземпляра типа. Любые поля, не задаваемые конструктором явно, гарантированно содержат 0 или null.

**9. Перечислите свойства конструктора?**

1) не имеет возвращаемого значения  
 2) имя такое же как и имя типа (класса)  
 3) не наследуются  
 4) нельзя применять модификаторы virtual, new,  
override, sealed и abstract  
 5) для класса без явно заданных конструкторов компилятор создает конструктор по умолчанию (без параметров)  
 6) для статических классов (запечатанных и абстрактных) компилятор не создает конструктор по умолчанию  
 7) может определяться несколько конструкторов, сигнатуры и уровни доступа к конструкторам обязательно должны отличаться  
 8) можно явно заставлять один конструктор вызывать другой конструктор посредством зарезервированного слова this:

**10. Что такое деструктор (destructor) ?**

Механизм очистки на уровне программного кода.

**Деструкторы** вызывается непосредственно перед окончательным  
уничтожением объекта системой "сборки мусора",  
чтобы гарантировать четкое окончание срока действия  
объекта.

**11. Что такое this?**

this обеспечивает доступ к текущему  
экземпляру класса  
 в любой нестатический метод автоматически  
передается скрытый параметр this

**12. Что будет выведено в результате выполнения**

5 5 7 7

**13. Какие спецификаторы доступа для класса и методов класса существуют в**

**C#?**

public, private, protected, internal

**14. Опишите модификатор protected internal.**

**protected –** (используется для вложенных классов) Доступен только методам в определяющем типе (и вложенных в него типах) или в одном из его производных типов независимо от сборки  
**internal** - доступ только из данной сборки

**15. Зачем и как используются ref и out параметры функции?**

При передаче параметров по ссылке перед параметрами используется модификатор ref

Чтобы сделать параметр выходным, перед ним ставится модификатор out

**16. Приведите пример необязательных и именованных параметров метода.**

Именованные

class Program

{

static int mySum(int a, int b = 5, int c = 10)

{

return a + b + c;

}

static void Main()

{

// Использование именованных аргументов

// при вызове метода

int sum1 = mySum(a: 3, b: 10);

Console.WriteLine("Sum1 = " + sum1);

Console.ReadLine();

}

}

Необязательные

class Program

{

// Аргументы b и с указывать при вызове необязательно

static int mySum(int a, int b = 5, int c = 10)

{

return a + b + c;

}

static void Main()

{

int sum1 = mySum(3);

int sum2 = mySum(3,12);

Console.WriteLine("Sum1 = "+sum1);

Console.WriteLine("Sum2 = "+sum2);

Console.ReadLine();

}

**17. Приведите пример полей класса – статические, константные, только для**

**чтения.**

private static int size;

private readonly int objHash = size + 1;

private const string month = "Октябрь";

**18. Приведите пример определения свойств класса. Как свойства связаны с**

**инкапуляцией?**

Как правило, свойство связано с закрытым полем класса и определяет методы его получения и установки (предоставляет инкапсуляцию).

public int Hour

{

get => hour;

set

{

if(value >= 0 && value < 24)

hour = value;

else

throw new Exception("Неверный формат времени.");

}

}

* **Назовите явное имя параметра, передаваемого в метод set свойства класса**

Value

* **Что такое автоматические свойства?**

Автоматическое свойство – это очень простое свойство, которое, в отличии от обычного свойства, уже определяет место в памяти (создает неявное поле), но при этом не позволяет создавать логику доступа. У таких свойств, у их аксессоров отсутствует тело.

* **Что такое индексаторы класса? Какие ограничения существуют наиндексатор?**

Индексаторы – это синтаксический инструмент языка C#, который позволяет индексировать внутренние коллекции типа. Индексаторы позволяют индексировать объекты и обращаться к данным по индексу. Фактически с помощью индексаторов мы можем работать с объектами как с массивами. По форме они напоминают свойства со стандартными блоками get и set, которые возвращают и присваивают значение.

* **Что такое перегруженный метод?**

Иногда возникает необходимость создать один и тот же метод, но с разным набором параметров. И в зависимости от имеющихся параметров применять определенную версию метода. Такая возможность еще называется перегрузкой методов (method overloading). И в языке C# мы можем создавать в классе несколько методов с одним и тем же именем, но разной сигнатурой.

* **Что такое partial класс и какие его преимущества?**

Классы могут быть частичными. То есть мы можем иметь несколько файлов с определением одного и того же класса, и при компиляции все эти определения будут скомпилированы в одно. Для этого они определяются с ключевым словом partial.

Преимущество: При работе над большим проектом, разбиение класса на несколько файлов позволяет нескольким программистам работать над ним одновременно.

* **Что такое анонимный тип в C#?**

Анонимные типы позволяют создать объект с некоторым набором свойств без определения класса. Анонимный тип определяется с помощью ключевого слова var и инициализатора объектов.

* **Для чего делают статические классы?**

Статические классы объявляются с модификатором static и могут содержать только статические поля, свойства и методы. Иногда требуется определить такой член класса, который будет использоваться независимо от всех остальных объектов этого класса. Как правило, доступ к члену класса организуется посредством объекта этого класса, но в то же время можно создать член класса для самостоятельного применения без ссылки на конкретный экземпляр объекта.

* **В чем отличие статического поля от экземплярного?**

Не нашел

* **Поясните работу статических конструкторов.**

Статический конструктор используется для инициализации любых [статических](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/static) данных или для выполнения определенного действия, которое требуется выполнить только один раз. Он вызывается автоматически перед созданием первого экземпляра или ссылкой на какие-либо статические члены.

* **Какая разница между поверхностным (shallow) и глубоким (deep)копированием?**

При поверхностном копировании копируются значения полей класса, включая значения любых указателей или ссылок. При этом скопированные значения этих указателей и ссылок указывают на одни и те же объекты, что и в оригинальном объекте, что зачастую ведет к ошибкам. Отсюда и название такого метода копирования: мы копируем только указатели/ссылки, вместо того, чтобы делать копии этих внутренних объектов и ссылаться на них, собственно не углубляемся во внутреннюю структуру объекта. При глубоком копировании мы копируем значения полей не только на первом "уровне", но и заходим глубже, копируя все значения.

* **В чем разница между равенством и тождеством объектов?**

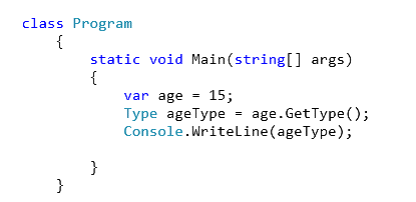
**Ответ не полный**

Метод Equals принимает в качестве параметра объект любого типа, который мы затем приводим к текущему, если они являются объектами одного класса. Затем сравниваем по именам. Если имена равны, возвращаем true, что будет говорить, что объекты равны.

* **Что такое частичные классы и частичные методы?**

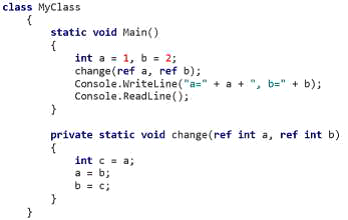
Классы могут быть частичными. То есть мы можем иметь несколько файлов с определением одного и того же класса, и при компиляции все эти определения будут скомпилированы в одно. Частичные классы могут содержать частичные методы. Таким методы также опреляются с ключевым словом partial. Причем определение частичного метода без тела метода находится в одном частичном классе, а реализация этого же метода - в другом частичном классе.

* **Что будет выведено на консоль результате выполнения следующего кода:**



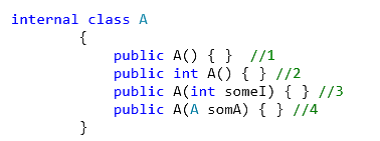
System.Int32

* **Что будет выведено на консоль результате выполнения следующего кода:**



a=2, b=1

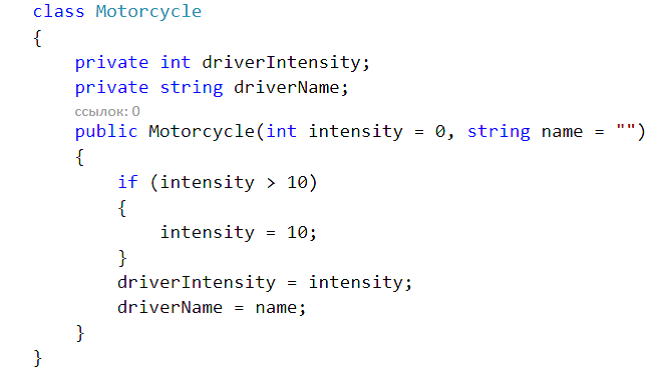
* **Пусть задан следующий класс.**



**Какой из конструкторов задан неверно?**

2, 4(не уверен про 4)

* **Пусть задан следующий класс.**

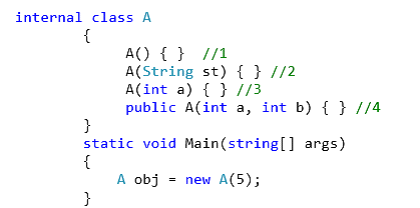


**Сколько аргументов может быть задано при вызове конструктора**

**данного класса?**

2

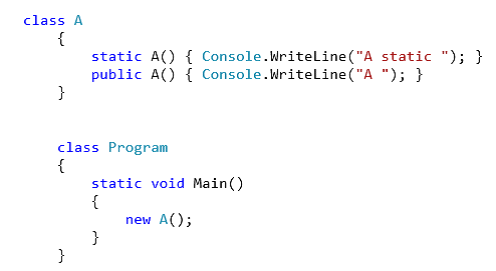
* **Почему не удается создать объект класса A?**



'Program.A.A(int)" недоступен из-за его уровня защиты.

(почему хз )

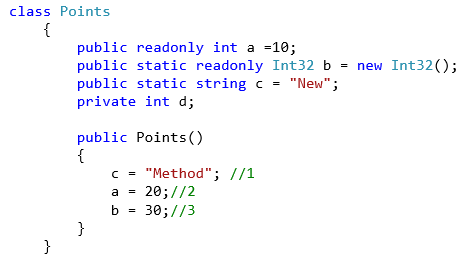
* **Что будет выведено в консоль при выполнении данной программы?**



A static

A

* **Какая строка приведенного далее класса вызовет ошибку компиляции?**



b=30; т.к. присвоение значения доступному только для чтения статическому полю допускается только в статическом конструкторе и в инициализаторе переменных.