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

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

2) Наследование (процесс, благодаря которому один объект может наследовать свойства другого объекта / иерархии классов)

3) Полиморфизм (способность вызывать метод потомка через экземпляр предка / способность программы идентично использовать объекты с одинаковым интерфейсом без информации о конкретном типе этого объекта)

4) Абстракция данных (абстракция подразумевает разделение и независимое рассмотрение интерфейса и реализации)

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

System.Object

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

public override bool Equals (object? obj) {...} // и возврашает true, если оба объекта равны

public static bool Equals (object ob1, object ob2);

public override int GetHashCode(); // возвращает хеш-код

person.GetType() // возвращает тип объекта

i.ToString() // возвращает строку

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

**5. Приведите пример определения класса.**

[ атрибуты ] [ спецификаторы ]

Class имякласса [ предок ]

{тело-класса }

class Book

{

public string name;

public string author;

}

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

New

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

Класс – шаблон, объект – сущность, собранная по шаблону

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

Конструкторы —это специальные методы, позволяющие корректно инициализировать новый экземпляр типа. Вызывается при создании объекта.

**9. Перечислите свойства конструктора?**

1) имя = имя типа (класса)

2) не имеет возвращаемого значения

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 (seald и abstract) 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)?**

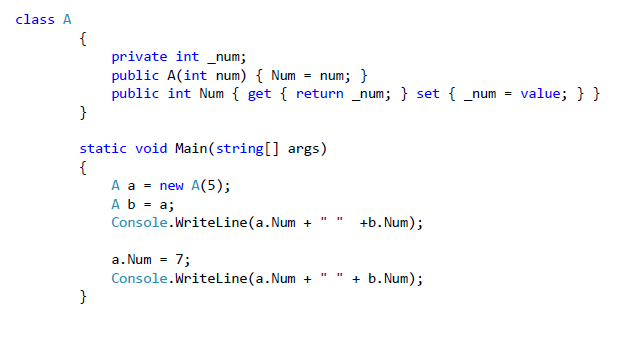
~имя\_класса () {код деструктора}

Дестру́ктор — специальный метод класса, служащий для деинициализации объекта. Вызывается перед окончательным уничтожением объекта системой "сборки мусора".

**11. Что такое this?**

Ключевое слово, обеспечивающее доступ к текущему экземпляру класса

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



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

Для классов:

* Private (только в рамках своего класса или структуры)
* Private protected (из любого места класса или производных класса (классы только в той же сборке))
* Protected (из любого места класса или производных класса (классы могут располагаться в разных сборках))
* Protected internal (в любом месте сборки и в производных в других сборках)
* Internal (в любом месте сборки)
* Public (везде)

**14. Опишите модификатор protected internal.**

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

Ref – параметры ссылки (вызов по значению меняет на вызов по ссылке)

Out – выходные параметры ссылки (можно использовать для передачи значения из метода)

[out-параметр "поступает" в метод без начального значения, но метод (до своего завершения) обязательно должен присвоить этому параметру значение]

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

Для необязательных параметров нам необходимо объявить значение по умолчанию. После необязательных параметров все последующие параметры также должны быть необязательными

void PrintPerson(string name, int age = 1, string company = "Undefined")

{ Console.WriteLine($"Name: {name} Age: {age} Company: {company}"); }

PrintPerson("Tom", 37, "Microsoft");  // Name: Tom  Age: 37  Company: Microsoft

PrintPerson("Tom", 37); // Name: Tom  Age: 37  Company: Undefined

PrintPerson("Tom"); // Name: Tom Age: 1 Company: Undefined

Именованные параметры помогают передавать аргументы не по позиции.

void PrintPerson(string name, int age = 1, string company = "Undefined")

{ Console.WriteLine($"Name: {name} Age: {age} Company: {company}"); }

PrintPerson ("Tom", company: "Microsoft", age: 37); // Name: Tom Age: 37 Company: Microsoft

PrintPerson (age:41, name: "Bob"); // Name: Bob Age: 41 Company: Undefined

PrintPerson (company: "Google", name: "Sam"); // Name: Sam Age: 1 Company: Google

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

class Person

{

public const string type = "Person";

public string name = "Undefined";

public readonly int age; // запись при объявлении или в виде конструктора

public void Print() => Console.WriteLine($"{type}: {name} - {age}");

}

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

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

set {действия, выполняемые при установке значения свойства}

value

**20. Что такое автоматические свойства?**

Свойства, в которых компилятор автоматически реализует методы для правильного возвращения значения из поля и назначения значения полю

public string Name { get; set; }

**21. Что такое индексаторы класса? Какие ограничения существуют на индексатор?**

Свойства с параметрами, которые позволяют индексировать объекты таким же способом, как массив или коллекцию

public Person this[int index]

{ get => personal[index];

set => personal[index] = value; }

Ограничения на индексаторы:

1)значение, выдаваемое индексатором, нельзя передавать методу в качестве параметра ref или out

2)индексатор не может быть объявлен как static

**22. Что такое перегруженный метод?**

23. Что такое partial класс и какие его преимущества?

24. Что такое анонимный тип в C#?

25. Для чего делают статические классы?

26. В чем отличие статического поля от экземплярного?

27. Поясните работу статических конструкторов.

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

29. В чем разница между равенством и тождеством объектов?

30. Что такое частичные классы и частичные методы?

31. Что будет выведено на консоль результате выполнения следующего кода:

32. Что будет выведено на консоль результате выполнения следующего кода:

33. Пусть задан следующий класс.

Какой из конструкторов задан неверно?

34. Пусть задан следующий класс.

Сколько аргументов может быть задано при вызове конструктора данного класса?

35. Почему не удается создать объект класса A?

36. Что будет выведено в консоль при выполнении данной программы?

37. Какая строка приведенного далее класса вызовет ошибку компиляции?

**Инкапсуляция -**скрытие некоторых моментов реализации класса от других частей программы.