Ответы на 3 лабу ООП

1) Назовите принципы ООП. Поясните каждый из них.

**Инкапсуляция** – механизм сохраняющий данные и код, от внешнего воздействия и ошибочного использования.

**Наследование** – процесс благодаря которому, один объект может наследовать свойства другого, позволяющий описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью.

**Полиморфизм** – способность вызывать метод наследника через объект предка.

**Абстракция** —  означает выделение значимой информации и исключение из рассмотрения незначимой. Это разделение программы на объекты. Абстракция позволяет отобрать главные характеристики и опустить второстепенные.

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

System. Object

3) Охарактеризуйте открытые методы System.Object.

Equals() - Сравнивает две ссылки на объекты в период выполнения, чтобы определить, указывают ли они в точности один и тот же объект. Если ссылаются на один и тот же объект, возвращается true.

GetHashCode() - Возвращает заданный для объекта хзш-код. Хэш-функции используются в реализации класса, когда хэш-код объекта нужно поместить в хэш-таблицу для повышения производительности

GetType() - Используется для получения информации о типе данного объекта.

ToString() - Используется по умолчанию для получения имени объекта. Его можно переопределить в производных классах, чтобы они возвращали понятное пользователю текстовое представление объекта.

4) Охарактеризуйте закрытые методы System.Object.

void Finalize() - Вызывается в период выполнения для освобождение ресурсов перед сбором мусора.

MemberwiseClone() - Представляет ограниченную копию (shallow copy) объекта. Копия объекта, содержащая ссылки на другие объекты, но не копии этих объектов. Если классы должны поддерживать полную копию (deep copy), которая включает к опии объектов, на которые она ссылается, то нужно реализовать интерфейс ICloneable и вручную производить клонирование или копирование.

5) Приведите пример определения класса.

Класс – абстрактное понятие (описывающая методы, свойства, ещё не существующих объектов), шаблон по которому определяется форма объекта.

6) Какие ключевые слова можно использовать при определении класса?

Абстракция, шаблон. Это описатель общих свойств группы объектов.

7) В чем отличие между объектом и классом?

Объект - экземпляр класса - объект, типом которого является какой-то класс. Объект после инициализации - сущность в памяти, обладающая поведением, способная изменять своё состояние.

8) Что такое конструктор? Когда вызывается конструктор?

Конструктор — это специальный метод, который вызывается при создании нового объекта. Конструктор инициализирует объект непосредственно во время создания. Имя конструктора совпадает с именем класса, включая регистр, а по синтаксису конструктор похож на метод без возвращаемого значения.

9) Перечислите свойства конструктора?

* Не имеет возвращаемого значения.
* Имя конструктора совпадает с именем класса.
* Не наследуется.
* Нельзя применить модификатор: virtual, new, abstract override и sealed.
* Для статических классов компилятор не создаёт конструктор по умолчанию.
* Может определяться несколько конструкторов и уровни доступа к конструкторам обязательно должны отличаться.
* Можно явно заставлять один конструктор вызывать другой с помощью зарезервированного слова this.

10) Что такое деструктор (destructor)?

Вызывается непосредственно перед окончательным уничтожением объекта системой «сборки мусора» чтобы гарантировать чёткое окончание срока действия объекта.

11) Что такое this?

This – зарезервированное слово, обеспечивающие доступ к текущему экземпляру класса. (В любой нестатический метод автоматически передаётся скрытый параметр this).

12) Что будет выведено в результате выполнения

10

12

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

Public, private, protected.

14) Опишите модификатор protected internal.

Доступно в любом месте программы (внутри одной сборки) и из классов наследников.

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

Ref – В C# указание модификатора ref приводит к созданию вызова по ссылке вместо вызова по значению. Данный модификатор указывается при объявлении метода и при его вызове.

Out – Модификатор используется для возврата более одного значения из метода. Отличие out от ref состоит в том что передаваемую переменную не обязательно инициализировать. Также метод объявленный с модификатором out обязан присвоить значение соответствующей переменной, до своего завершения и передачи управления вызывающей функции.

16. Приведите пример необязательных и именованных параметров метода.

**Необязательные аргументам** позволяют определить используемое по умолчанию значение для параметра метода. Данное значение будет использоваться по умолчанию в том случае, если для параметра не указан соответствующий аргумент при вызове метода.

*// 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);*

*}*

**Именованный аргумент** позволяет указать имя того параметра, которому присваивается его значение. И в этом случае порядок следования аргументов уже не имеет никакого значения.

*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);*

*}*

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

*public static int x1;*

*private const int x2;*

*protected readonly x3;*

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

*private int phone {get; set;} //автоматическое свойство*

[модификатор\_доступа] возвращаемый\_тип произвольное\_название

*{*

*// код свойства*

*}*

Через свойство можно управлять доступом к переменной name. В блоке get мы возвращаем значение поля, а в блоке set устанавливаем. Параметр value представляет передаваемое значение.

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

Параметр value представляет передаваемое значение.

20. Что такое автоматические свойства?

*private int phone {get; set;}*

Также создаются поля для свойств, только их создает не программист в коде, а компилятор автоматически генерирует при компиляции. Стандартные свойства имеют ряд преимуществ: например, они могут инкапсулировать дополнительную логику проверки значения. Нельзя создать автоматическое свойство только для записи, как в случае со стандартными свойствами.

21. Что такое индексаторы класса? Какие ограничения существуют на индексаторе?

Свойства с параметрами. Позволяют индексировать объекты также, как и массивы или коллекции. (позволяют перегружать оператор [] ). Индексаторы могут быть одно- или многомерными.

*атрибуты спецификаторы тип this [список\_ параметров]*

*{*

*get //код доступа*

*}*

*{*

*set //код доступа*

*}*

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?

**Перегрузка методов** – это совместное использование одного и того же имени двумя или более методами одного и того же класса, при условии, что их параметры объявляются по-разному. Перегрузка методов относится к одному из способов реализации полиморфизма в C#.

23. Что такое partial класс и какие его преимущества?

Частичные классы **(partial class)** представляют возможность разделить функционал одного класса на несколько файлов. Предусматривается использование шаблонов и контроля версий.

24. Что такое анонимный тип в C#?

Анонимные типы позволяют создать объект с некоторым набором свойств без определения класс.

*var имя = new {параметр}*

25. Для чего делают статические классы?

Статический класс требуется при создании *метода расширения*. Методы расширения связаны в основном с языком LINQ.

Статический класс служит для хранения совокупности связанных друг с другом статических методов.

26. В чем отличие статического поля от экземплярного?

Если определение поля содержит модификатор **Shared** поле называется статическим. Когда Shared нет в определении поля то, оно экземплярное. Статические и экземплярные поля являются двумя разными видами переменных, поддерживаемых языком.

Статическое поле идентифицирует ровно одну ячейку памяти. Вне зависимости от того какое количество экземпляров класса создано существует только одна копия статического поля в домене приложения.

Экземплярное поле относится к определенному экземпляру. Каждый экземпляр класса содержит отдельный набор всех экземплярных полей данного класса.

27. Поясните работу статических конструкторов.

Конструктор экземпляра инициализирует данные экземпляра.

Статический конструктор нельзя вызывать напрямую. Если обычный конструктор вызывается в момент создания экземпляра класса, то статический – вызывается автоматически перед созданием первого экземпляра класса или перед использованием ссылки на какие-либо статические члены.

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

(shallow copy) Копия объекта, содержащая ссылки на другие объекты, но не копии этих объектов.

(deep copy) Если классы должны поддерживать полную копию, которая включает копии объектов, на которые она ссылается, то нужно реализовать интерфейс ICloneable и вручную производить клонирование или копирование.

29. В чем разница между равенством и тождеством объектов?

[Сравнение через  ==](https://msdn.microsoft.com/ru-ru/library/53k8ybth.aspx). Это сравнение разрешается *на этапе компиляции* согласно заявленным типам левой и правой части сравнения. Оператор сравнения можно перегрузить. Он является статическим, настоящий тип объектов не играет роли при выборе того, какой из перегруженных операторов сравнения будет использован.

[Equals(object)](https://msdn.microsoft.com/en-us/library/bsc2ak47%28v=vs.110%29.aspx). Это сравнение разрешается вызовом виртуальной функции Equals и основывается на *динамическом* типе левого аргумента

30. Что такое частичные классы и частичные методы?

Частичные классы **(partial class)** представляют возможность разделить функционал одного класса на несколько файлов. Предусматривается использование шаблонов и контроля версий.

Частичные методы:

* Не могут использовать out.
* Используются внутри частичного класса или структуры.
* Должны всегда иметь тип void.
* Может иметь параметр ref, unsafe, универсальные параметры, статические и экземплярные.
* Private не пишется.

31) анонимный тип

32) 1 2

33) 4

34) 2

35) private

36) A static

37) 2