***Ответы на вопросы к лабораторной работе №3***

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

Принципы ООП:

* Инкапсуляция – механизм, связывающий вместе данные и код, обрабатывающий эти данные, и сохраняющий их от внешнего воздействия и ошибочного использования.
* Наследование - процесс, благодаря которому один объект может наследовать свойства от другого объекта.
* Полиморфизм – способность вызывать метод потомка через экземпляр предка.
* Абстракция данных – разделение и независимое рассмотрение интерфейса и реализации.

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

*System.Object.*

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

Открытые (которые можно переопределять )методы System.Object: ToSring(), Equals() и ReferenceEquals(), GetType(), GetHashCode()

**ToSring():** Служит для получения строкового представления объекта. Для классов - выводит полное название класса с указанием пространства имён, в котором определён этот класс.

**Equals() и ReferenceEquals() :** Метод Equals позволяет сравнить два объекта на равенство. По умолчанию метод Equals (object) определяет, ссылается ли вызывающий объект на тот же самый объект, что и объект, указываемый в качества аргумента этого метода, т.е. он определяет, являются ли обе ссылки одинаковыми. Метод Equals (object) возвращает логическое значение true, если сравниваемые объекты одинаковы, в противном случае — логическое значение false. Если есть переопределение – метод Equals (object) не может использоваться для проверки на тождественность, но может быть использован для проверки на равенство. ReferenceEquals() – тождественнное сравнение ссылок.

**GetHashCode():** Метод GetHashCode позволяет возвратить некоторое числовое значение, которое будет соответствовать данному объекту или его хэш-код. По данному числу, например, можно сравнивать объекты.

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

**Закрытые(которые нельзя переопределять**) методы System.Object: Finalize(), GetType(), Clone()

**GetType():**Этот метод возвращает экземпляр класса, унаследованный от System.Туре. Этот объект может предоставить большой объем информации о классе, членом которого является ваш объект, включая базовый тип, методы, свойства и т.п. System.Туре также представляет собой стартовую точку технологии рефлексии .NET.

**Finalize():**Назначение этого метода в С# 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. Сборщик мусора не может сделать это напрямую, потому что он знает только об управляемых ресурсах, поэтому полагается на финализацию, определенную вами.

**Clone():**Этот метод создает копию объекта и возвращает ссылку на эту копию. При этом выполняется поверхностное копирование, т.е. копируются все значения в классе. Если же класс включает в себя члены ссылочных типов, то копируются только ссылки, а не объекты, на которые они указывают. Этот метод является защищенным, а потому не может вызываться для копирования внешних объектов. К тому же он не виртуальный, а потому переопределять его реализацию нельзя.

**5.Приведите пример определения класса.**

**public class Student**

**{**

**public string name;**

**public string secondName;**

**public int course;**

**public void Info()**

**{**

**Console.WriteLine($”{name} {course}”);**

**}**

**}**

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

Класс определяется с помощью ключевого слова сlass:

Ключевое слово namespace определяет пространство имён, в котором определён класс.

[модификатор доступа] class [имя\_класса]  
{  
   //тело класса  
}

Модификатор доступадля класса :

-public-доступ к классу возможен из любого места одной сборки либо из другой сборки, на которую есть ссылка;

-internal-доступ к классу возможен только из сборки, в которой он объявлен.

namespace HelloWorld  
{  
   class Student //без указания модификатор доступа, класс будет internal  
   {  
     //тело класса  
   }  
   public class Pupil  
  {  
    //тело класса  
  }  
}

Уровни доступа для членов класса:  
- public – доступ к члену возможен из любого места одной сборки, либо из другой сборки, на которую есть ссылка;  
- protected – доступ к члену возможен только внутри класса, либо в классе-наследнике (при наследовании);  
- internal – доступ к члену возможен только из сборки, в которой он объявлен;  
- private – доступ к члену возможен только внутри класса;  
- protected internal - доступ к члену возможен из одной сборки, либо из класса-наследника другой сборки.

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

Класс-это шаблон, по которому определяется объект.

Объект-физическая реализация класса.

Объекты создаются и уничтожаются по мере необходимости, поэтому их время жизни ограничено, в отличии от класса.

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

**Конструктор** – это метод класса, предназначенный для инициализации объекта при его создании.  Конструктор автоматически вызывается на этапе компиляции при создании экземпляра данного класса. Попытки вызвать конструктор явным образом вызовет ошибку компиляции.

**9.Перечислите свойства конструктора.**

**Свойства конструктора:**

* Не имеет возвращаемого значения;
* Имя конструктора совпадает с именем класса, в котором он объявлен;
* Не наследуется;
* Нельзя применять модификаторы virtual, new, override, sealed, abstract.
* Для класса без явно заданных конструкторов компилятор создаёт конструктор по умолчанию (без параметров);
* Для статических классов компилятор не создаёт конструктор по умолчанию;
* Может определяться несколько конструкторов, но сигнатуры и уровни доступа к конструкторам обязательно должны отличаться;
* Можно явно заставлять один конструктор вызывать другой конструктор посредством зарезервированного слова this;

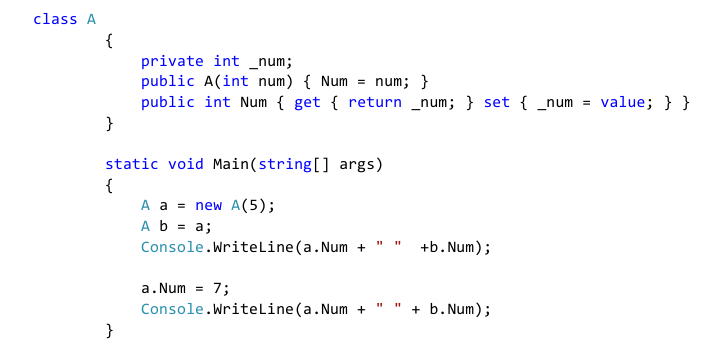
**10.Что такое деструктор?**

**Деструктор –** метод, вызывающийся непосредственно перед окончательным уничтожением объекта системой «сборки мусора».

**11.Что такое this?**

This – ключевое слово, обеспечивающее доступ к текущему экземпляру класса.

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



5 5 7 7

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

Спецификаторы доступа для класса :

-public-доступ к классу возможен из любого места одной сборки либо из другой сборки, на которую есть ссылка;

-internal-доступ к классу возможен только из сборки, в которой он объявлен.

Спецификаторы доступа для методов класса:  
- public – доступ к члену возможен из любого места одной сборки, либо из другой сборки, на которую есть ссылка;  
- protected – доступ к члену возможен только внутри класса, либо в классе-наследнике (при наследовании);  
- internal – доступ к члену возможен только из сборки, в которой он объявлен;  
- private – доступ к члену возможен только внутри класса;  
- protected internal - доступ к члену возможен из одной сборки, либо из класса-наследника другой сборки.

**14. Опишите модификатор protected internal ?**

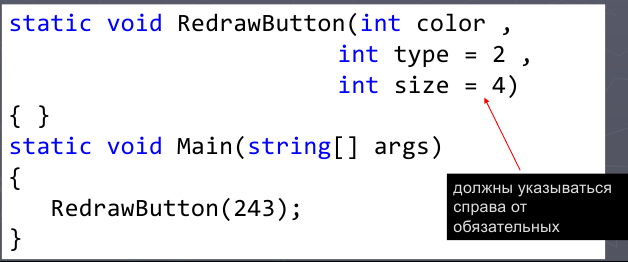
Protected internal  является модификатором доступа к члену и делает возможным доступ к члену из одной сборки, либо из класса-наследника другой сборки

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

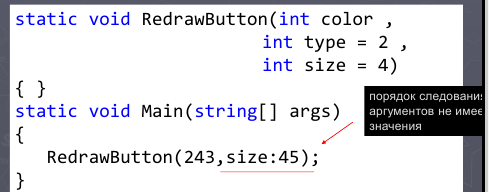
Использование ref и out приводит к тому, что параметр начинает передаваться не по значению, а по ссылке. Разница между ними в том, что **out** позволяет передать ссылку на переменную неинициализированную, а **ref** — только на инициализированную, т.е. аргументу, передаваемому методу “в сопровождении” модификатора ref, должно быть присвоено значение до вызова метода, а out-параметр «поступает» в метод без начального значения, но метож обязательно должен присвоить этому параметру значение.

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

Необязательные параметры(в примере type и size ) можно их опускать



Именованные параметры (в примере size ) –позволяют нарушать порядок объявления параметров.



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

**Пример статического поля класса**

public static decimal bonus = 100;

**Пример константного поля класса**

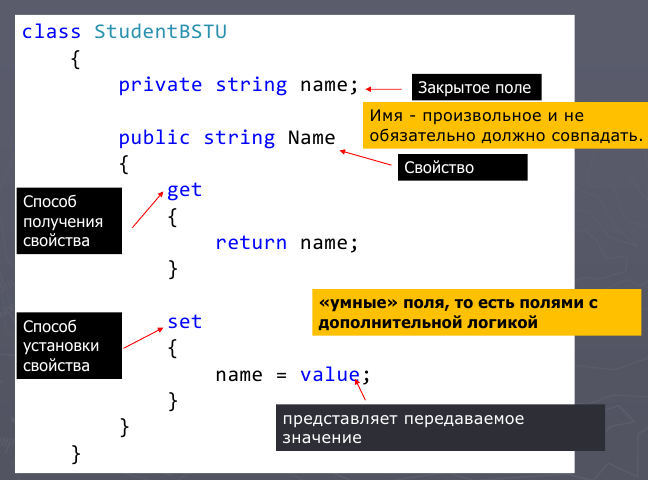
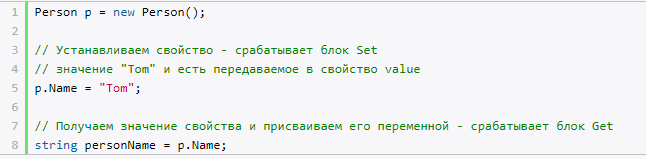
public const double PI=3.141;

**Пример поля только для чтения**

public readonly double K = 23;

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

Определение свойств класса осуществляется через блоки get и set.



Инкапсуляция позволяет скрывать некоторые моменты реализации класса от других частей программы.

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

**Value.** Тип этого параметра определяется типом свойства.

**20.Что такое автоматические свойства?**

**Автоматические свойства-** свойства, которые создают неявные поля, но при этом не позволяют создавать логику доступа.

Компилятор автоматически реализует методы для правильного возвращения значения из поля и назначения значения полю**21.Что такое индексаторы класса? Какие ограничения существуют на индексатор?**

**Индексатор – это средство языка C#,** позволяющее индексировать объект так как массив с помощью прямоугольных скобок[].

**Ограничения на индексаторы;**

1. Значение, выдаваемое индексатором, нельзя передавать методу в качестве параметра out или ref;
2. Индексатор не может быть объявлен как static

**22.Что такое перегруженный метод?**

Перегруженным методом называется несколько методов определённых в одном классе ,с одинаковым именем, но с разным набором параметров.

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

**Partial класс (частичный класс) –**модификатор , используемый для разбития определения класса, структуры или интерфейса между двумя или большим количеством исходных файлов. Каждый исходный файл содержит свою часть определения класса и все такие части собираются во время компиляции.

**Преимущества partial класса:**

1. При работе над большим проектом, разбиение класса на несколько файлов позволяет нескольким программистам работать над ним одновременно.  
2. При работе с автоматически генерируемыми исходниками, код может быть добавлен в класс без необходимости пересоздавать исходник. Visual Studio использует этот подход при создании компонентов Windows Forms, веб-сервисов и т.д. Возможно создать код, который использует эти классы без необходимости редактировать файлы, которые создает Visual Studio.

**24.Что такое анонимный тип в С#?**

Анонимный тип – это возможность, позволяющая создавать объект с некоторым набором свойств без определения класса.

**25. Для чего делают статические классы?**

Статический класс в основном такой же, как и нестатический класс, но имеется одно отличие: нельзя создавать экземпляры статического класса. Другими словами, нельзя использовать ключевое слово new для создания переменной типа класса. Поскольку нет переменной экземпляра, доступ к членам статического класса осуществляется с использованием самого имени класса.Статический класс может использоваться как обычный контейнер для наборов методов, работающих на входных параметрах, и не должен возвращать или устанавливать каких-либо внутренних полей экземпляра.

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

Когда определение поля содержит модификатор Shared такое поле называется статическим. Когда модификатора Shared нет в определении поля оно называется экземплярным. Статические и экземплярные поля являются двумя разными видами переменных, поддерживаемых языком, и временами называются статическими и экземплярными переменными соответственно.

Статическое поле не является частью определенного экземпляра, наоборот, оно 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.**

Статический конструктор используется для инициализации любых статических 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.

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

Поверхностное копирование работает только для свойств, представляющих примитивные типы, но не для сложных объектов, а **глубокое копирование** работает и для сложных объектов.

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.

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

Тождество - равенство, выполняющееся при любых значениях аргумента(ов) и одинаковых областях допустимых значений этого аргумента(ов) слева и справа от знака "=".  
Равенство же не обязано соответствовать этим двум условиям - может или ОДЗ слева и справа разные или не при любых значениях аргументов равенство выполняется (может даже ни при одном не выполняться).

**30. 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. Для этого они определяются с ключевым словом partial.

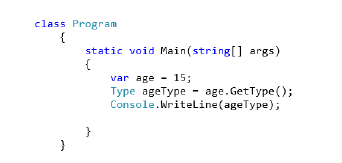
Частичные классы могут содержать частичные методы. Таким методы также опреляются с ключевым словом partial. Причем определение частичного метода без тела метода находится в одном частичном классе, а реализация этого же метода - в другом частичном классе.

При этом частичные методы не могут иметь модификаторов доступа - по умолчанию они все считаются приватными. Также частичные методы не могут иметь таких модификаторов как virtual, abstract, override, new, sealed. Хотя допустимы статические частичные методы.

Кроме того, частичные методы не могут возвращать значения, то есть они всегда имеют тип void. И также они не могут иметь out-параметров.

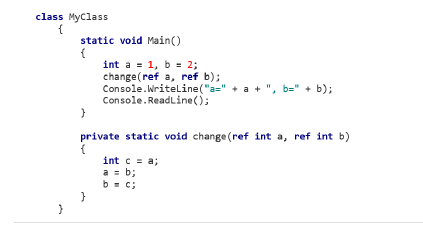
Поскольку частичные методы всегда приватные, то мы не сможем из вызвать напрямую в программе вне классов, где они определены.

**31. Что будет выведено на консоль в результате выполнения следующего кода?**



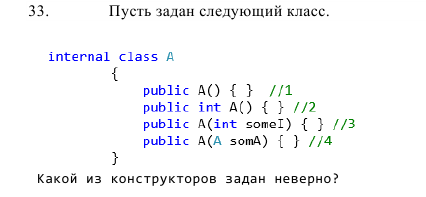
System.Int32

**32. Что будет выведено на консоль результате выполнения следующего кода?**



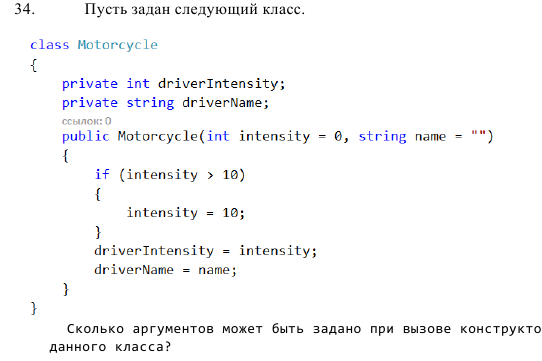
a=2, b=1

**33. Пусть задан следующий класс. Какой из конструкторов задан неверно?**



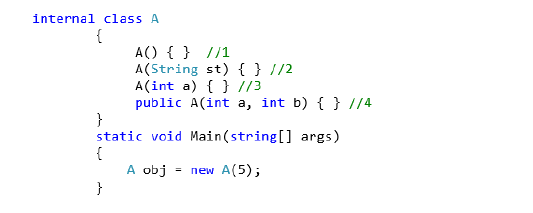
Неверна 2 строка, т.к. в конструктор не может возвращать значения

**34. Пусть задан следующий класс. Сколько аргументов может быть задано при вызове конструктора данного класса?**



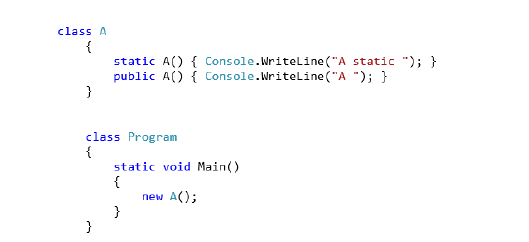
2

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



В первой строке нет слова public и не содержит аргумента Во второй строке нету слова public и содержит не тот тип аргумента который передается В третьей строке нету слова public В четвертой строке определено 2 аргумента, а требуется 1

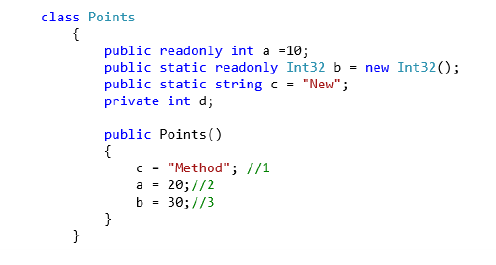
**36. Что будет выведено в консоль при выполнении данной программы?**



A static

A

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



2 строка выведет ошибку, так как переменной для чтения уже присвоено значение