1. **Для чего используют статические классы?**

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. Например, в библиотеке классов .NET статический класс System.Math содержит методы, выполняющие математические операции, без требования сохранять или извлекать данные, уникальные для конкретного экземпляра класса Math. Это значит, что члены класса применяются путем задания имени класса и имени метода.

1. **Что может содержать статический класс?**

Статические классы объявляются с модификатором static и могут содержать только статические поля, свойства и методы.

– Содержит только статические члены.   
– Создавать его экземпляры нельзя.   
– Является запечатанным.   
– Не может содержать конструкторы экземпляров.

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.

1. **Как используют ключевое слово base?**

Ключевое слово base используется для доступа к членам базового из производного класса в следующих случаях:   
– Вызов метода базового класса, который был переопределен другим методом.   
– Определение конструктора базового класса, который должен вызываться при создании экземпляров производного класса.   
Доступ к базовому классу разрешен только в конструкторе, методе экземпляра или методе доступа к свойству экземпляра.   
Использование ключевого слова base в статическом методе является недопустимым.

1. **В чем заключена основная задача наследования?**

Основная задача заключается в получении нового класса на основе существующего.

1. **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).

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

public B : base()

{

//…

//...

}

1. **Что такое полиморфизм? Приведите пример**

Полиморфизм – это взаимозаменяемость объектов с одинаковые интерфейсом. Полиморфизм – это один из столпов ооп, заключается он во фразе "один шаблон, множество реализаций" как пример, переопределение методов, свойств, полей, конструкторов.

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

1. **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.

1. **Кому доступны переменные с модификатором protected?**

Любым классам-наследникам.

1. **Наследуются ли переменные с модификатором private?**

Да, но они не являются доступными.

1. **As, is – что это, как применяется? В чем между ними отличие ?**

Ключевое слово AS служит для приведения объекта к указанному типу

В случае невозможности привести объект к указанному типу мы вместо исключения получим null.

Ключевое слово IS позволяет сравнить объект и тип данных, чтобы определить принадлежит ли наш объект этому типу. Возвращает данная конструкция либо true либо false.

1. **Поддерживает ли C# множественное наследование?**

Не поддерживает.

1. **Можно ли запретить наследование от класса?**

Да с помощью sealed.

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

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

1. **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.

1. **В каком случае вы обязаны объявить класс абстрактным?**

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

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

НЕТУ ВИРТУАЛЬНЫХ КЛАССОВ

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.

1. **Какие компоненты класса могут быть виртуальными?**

Методы, свойства, индексаторы, события.

1. **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.**

1. **Что может содержать 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.**

1. **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).

1. **Приведите пример явной реализации интерфейса.**

Public interface IName

{

Void WriteName();

}

Public interface IFamily

{

Void WriteName();

}

1. **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.**

1. **Можно ли наследовать от нескольких интерфейсов?**

Можно**.**

1. **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.

Если надо определить общий функционал для родственных объектов.

Если нам надо определить функционал для группы разрозненных объектов, которые могут быть никак не связаны между собой.

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

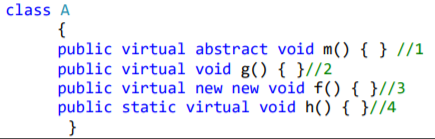
1) для клонирования обьекта

2) для сравнения обьектов(2 параметра в методе)

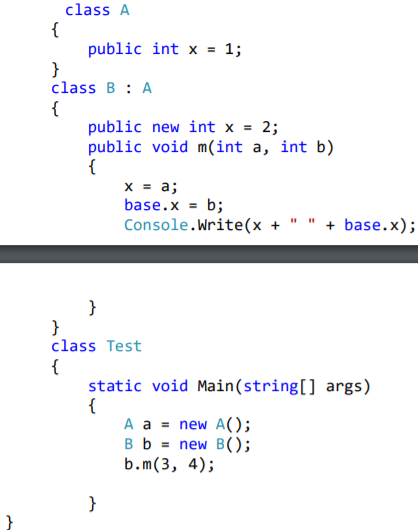
3)тоже самое только другая форма

4)для структурированной коллекции и цикла for each

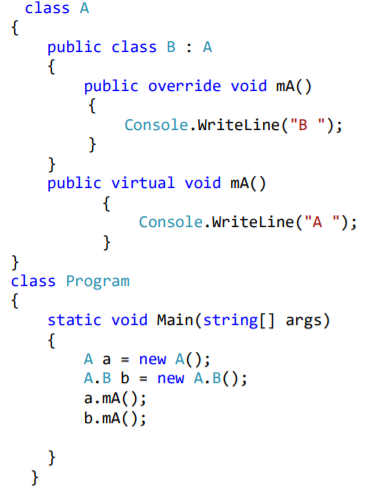
1. **В какой строке приведенного ниже фрагмента листинга не содержится ошибки? – 2**

****

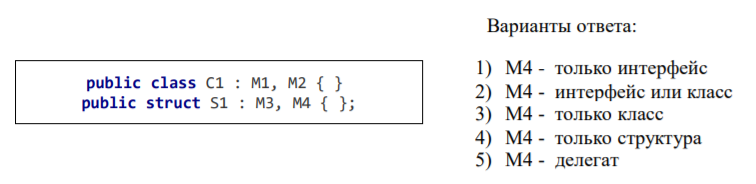
1. **Что будет выведено на консоль в результате выполнения следующего фрагмента? – 3,4**

****

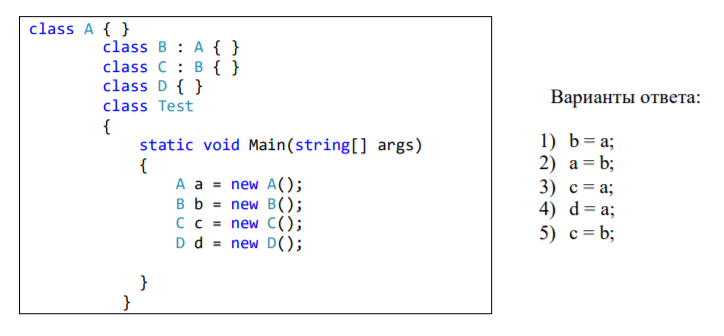
1. **Что будет выведено на консоль в результате выполнения следующего фрагмента кода? – A B**

****

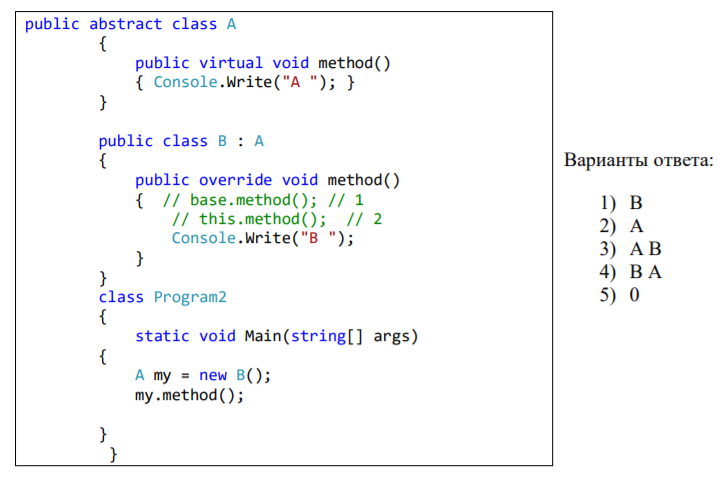
1. **Чем может быть M4 если дано следующее определение: – 1**

****

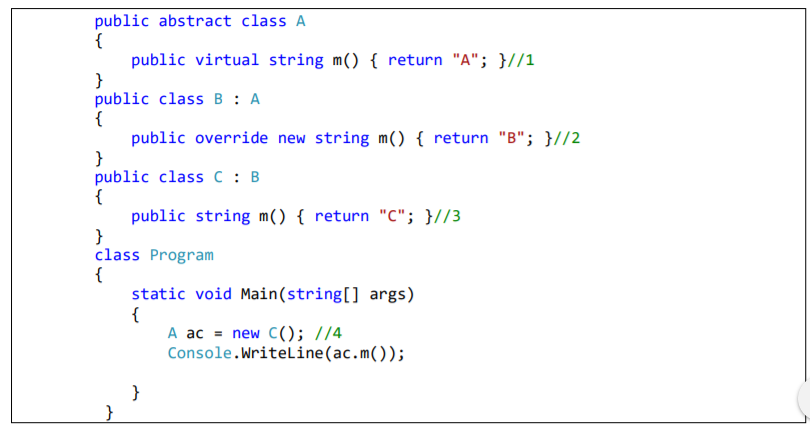
1. **. Выберите верное присваивание для объектов, определенных в листинге. – 2**

****

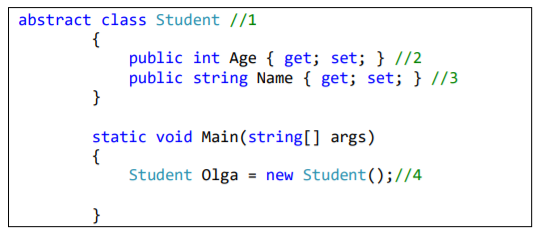
1. **Что будет выведено на консоль в результате выполнения следующего фрагмента, если раскомментировать строчку 1? – 3**

****

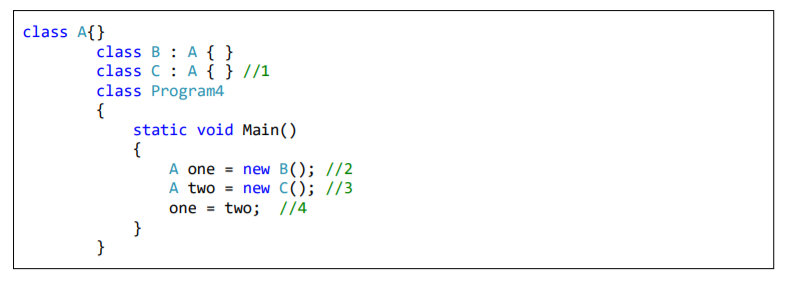
1. **В какой строке приведенного ниже фрагмента листинга содержится ошибка? – 2**

****

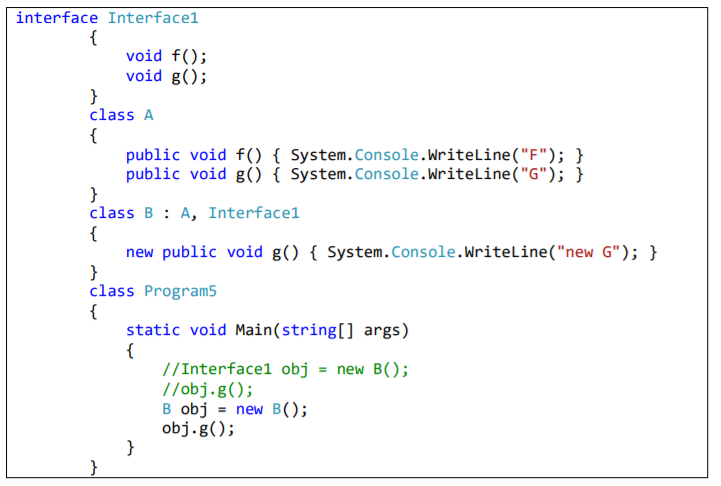
1. **Почему приведенный ниже фрагмент листинга содержит ошибку? – 4 (через абстрактный класс нельзя создать объект)**

****

1. **В какой строке может быть ошибка компиляции? – ошибки нет??**

****

1. **Что будет выведено на консоль в результате выполнения следующего фрагмента листинга: – ошибка класс B реализует интерфейс но не переопределяет все его методы**

****