1) Для чего используют статические классы

1) при создании метода расширения (позволяют добавлять новые методы в уже существующие типы без создания нового производного класса)

2) для хранения совокупности связанных друг с другом статических методов

2) Что может содержать статический класс?

1. экземпляры такого класса создавать запрещено(можно через сам класс)
2. не должен реализовывать никаких интерфейсов (не вызвать)
3. нельзя использовать в качестве поля, параметра метода или локальной переменной
4. от него запрещено наследовать
5. все элементы такого класса должны явным образом объявляться с модификатором static (поля, методы, свойства)
6. может иметь статический конструктор
7. компилятор не создает автоматически конструктор по умолчанию

3) Что такое производный и базовый классы?

Базовый класс – уже существующий класс.

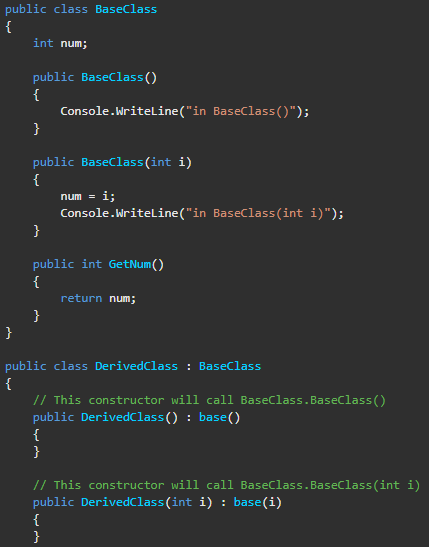
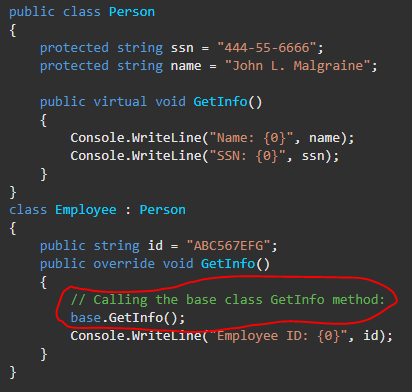
**Производный класс** – класс, который может наследовать свойства и методы базового класса.

4) Как используют ключевое слово base? Ключевое слово **base** используется в производном классе для обращения к членам базового класса:

* 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 недопустимо.

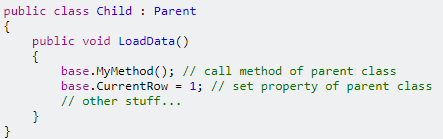


5) В чем заключена основная задача наследования?

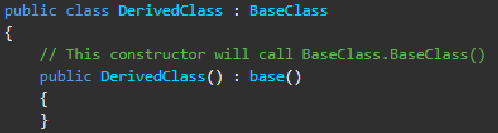
**Наследование** — один из принципов ООП. Механизм создания нового класса на основе уже существующего. Формирует иерархию и поощряет повторное использование кода.

6) 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, при обычном наследовании (если protected или public или internal)!!! если basefunc() не private



7) Напишите объявление конструктора без аргументов для производного класса В, который будет вызывать конструктор без аргументов базового класса А.



class A  
 public A() {} *//к-ры не наслед.*  
 public A(int key) {}  
class B : A  
 public B(int key) {}  
 public B() : ***base***(125) {}  
void main()  
 B b0 = new B(); *//авто- вызов к-р базового класса без парам*  
 B b1 = new B(125);

8) Что такое полиморфизм? Приведите пример.

►Полиморфизм (Polymorphism) — позволяет разным сущностям выполнять одни и те же действия. Полиморфизм позволяет повысить коэффициент повторного использования кода. Программист может сделать два разных метода — один для картинки, другой для видео. А может воспользоваться полиморфизмом и создать один метод «Отправить» для обеих сущностей.

Статический – перегрузка методов и операторов.

Динамический – наследование, реализация интерфейсов, переопределение методов

Один интерфейс, множество реализаций.

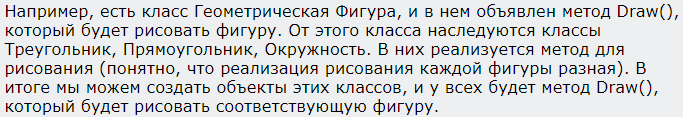
- способность вызывать метод потомка через экземпляр предка

- способность программы идентично использовать объекты с одинаковым интерфейсом без информации о конкретном типе этого объекта

- переопределение членов баз. класса

**virtual** public void A\_method() {}  
переопределение виртуального метода в произв. классе:  
**override** public void A\_method() {}

вызов метода по типу ссылки *//просто*  
вызов метода по типу объекта *//virtual + override*



9) Определите назначение виртуальных функций.

**Виртуальная функция** — это функция-член, которую предполагается переопределить в производных классах. Реализуется в базовом классе. Они позволяют реализовать полиморфизм, позволяя объектам разных классов вызывать одинаково названные методы.

**override** – изменение аб. или вирт. реализации  
**virtual** – разрешает переопределение (↑)

переопределение:  
\* те же парам.  
\* не static, не abstract

10) Кому доступны переменные с модификатором protected?

Базовому и производным классам из текущей или другой сборки (любому классу-наследнику)

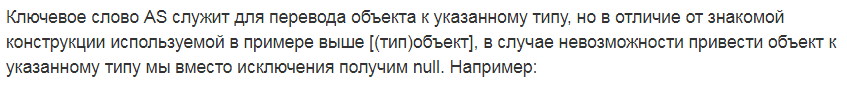
11) Наследуются ли переменные с модификатором private?

Нет (да, но они не являются доступными)

12) As, is – что это, как применяется?

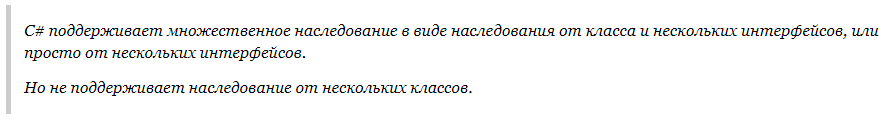
**Is – можно преобразовать данное выр. в указ. тип? (t/f)**

int j = 12  
object boxed = j  
object obj = new Object()  
bool a = boxed is int //true  
bool b = obj is int //false



**As – позв. преобр. тип в опр. ссылочный тип:**операнд as <тип> //если оп-д имеет <тип>, мб неявно-, мб упакован

13) Поддерживает ли С# множественное наследование?



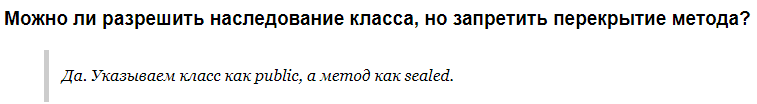
14) Можно ли запретить наследование от класса?

**sealed** – запрещает другим классам наследовать от данного  
*sealed class B : A {} – класс В наследует от класса А, но никакие не могут насл. от В*при применении к методу/свойству, исп. OVERRIDE

15) Можно ли разрешить наследование класса, но запретить перекрытие метода?

**Да, можно**. Пример:

class A  
 public virtual int Sum()  
 { return 2 + 3; }  
class B : A  
 public **sealed override** int Sum() *//метод в незапеч. классе явл. запеч.* { return 2 + 5; }

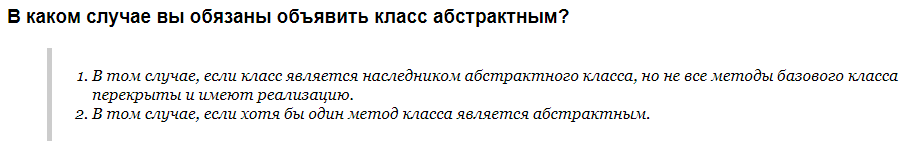


В C# можно разрешить наследование класса, но запретить переопределение (перекрытие) метода, используя ключевое слово **sealed** перед методом, который не должен быть переопределен.

16) Что такое абстрактный класс?

**Абстрактный класс** — класс, объект которого не мб создан и предназначен для служения в качестве базового класса для других классов. Абстрактный класс может содержать (и не содержать) абстрактные (не реализованные) методы. Такой класс должен иметь производный класс с реализацией абстрактных методов. Абстрактные классы реализуют полиморфизм. Абстрактный класс – это фактически чертёж нормального класса без реализации.  
\* м. опред. методы, пер., к-ры, св-ва

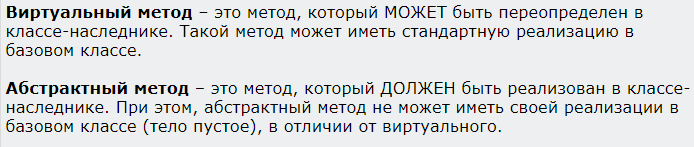
abstract class Shapes  
 abstract public int Area();  
  
class Square : Shapes {  *//д. обесп. реализацию Area* public override int Area()17) В каком случае вы обязаны объявить класс абстрактным?



Класс должен быть объявлен абстрактным, если он содержит один или несколько абстрактных методов. Если он не предполагает создания экземпляров и должен использоваться только в качестве базового класса.

18) В чем разница между абстрактными и виртуальными классами? Между виртуальными и абстрактными методами?

* Абстрактный класс не может быть инстанциирован, в то время как виртуальный класс может.
* Абстрактный класс может содержать абстрактные методы (методы без реализации), которые должны быть переопределены в производных классах, в то время как виртуальные методы виртуальных классов имеют реализацию по умолчанию, которую можно переопределить в производных классах.



19) Какие компоненты класса могут быть виртуальными?

методы, свойства, индексаторы, события

20) Что такое интерфейс?

**Интерфейсы**, как и классы, определяют набор свойств, методов и событий. Но, в отличие от классов, они не содержат их реализации. Интерфейсы реализуются классами и определяются как самостоятельные сущности.

Interface IDo  
 void Go()  
 int Jumn (int a)  
 void Sleep()  
 int Energy { get; }

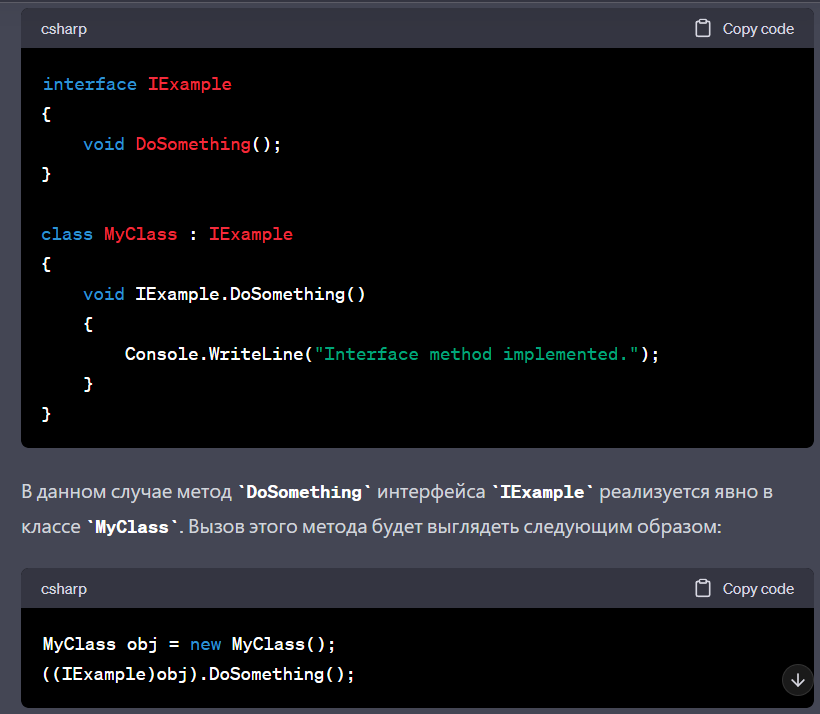
21) Что может содержать интерфейс?

1. Свойства
2. Методы
3. Индексаторы
4. События
5. Статические поля и константы

22) Как работать с объектом через унаследованный интерфейс?

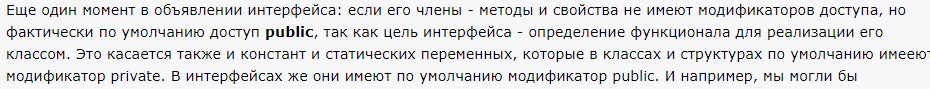
Для работы с объектом через унаследованный интерфейс, вы должны создать экземпляр объекта и присвоить его переменной, тип которой является интерфейсом. Затем вы можете вызывать методы, использовать свойства и обрабатывать события через этот интерфейс.

23) Приведите пример явной реализации интерфейса.



24) Почему нельзя указать модификатор видимости для методов интерфейса?

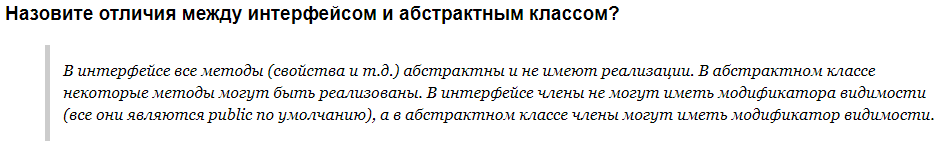
Потому что все они должны иметь модификатор ***public***, который и установлен по умолчанию.



25) Можно ли наследовать от нескольких интерфейсов?

Да, через запятую.

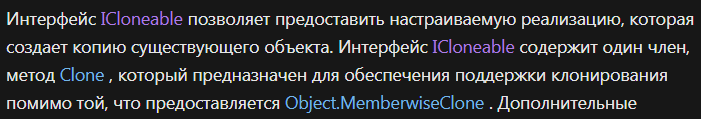
26) Назовите отличия между интерфейсом и абстрактным классом.

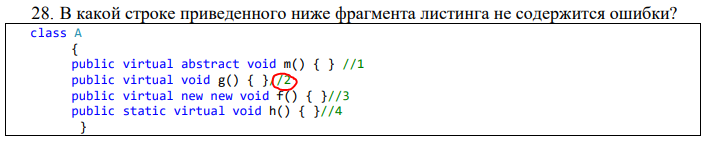


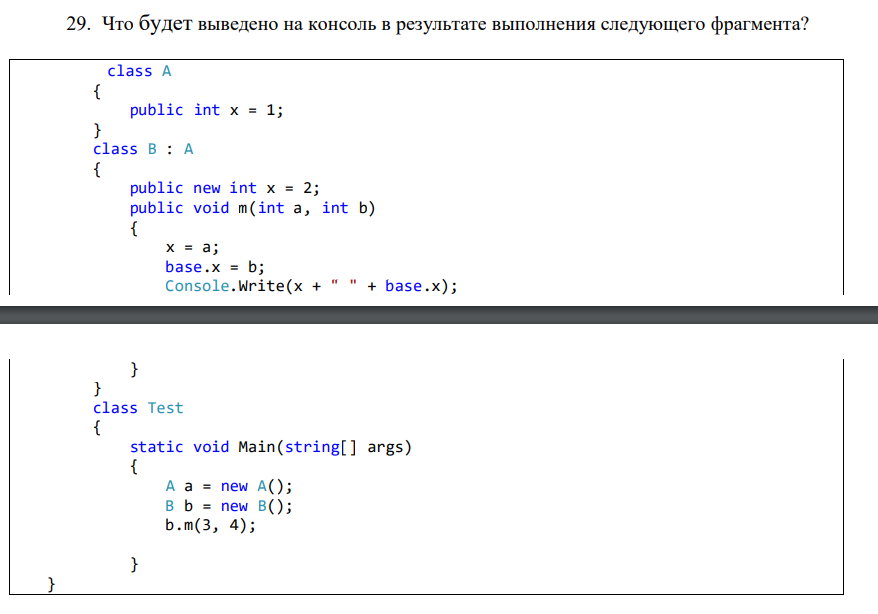
Класс может наследовать только один абстрактный класс, но реализовывать множество интерфейсов.



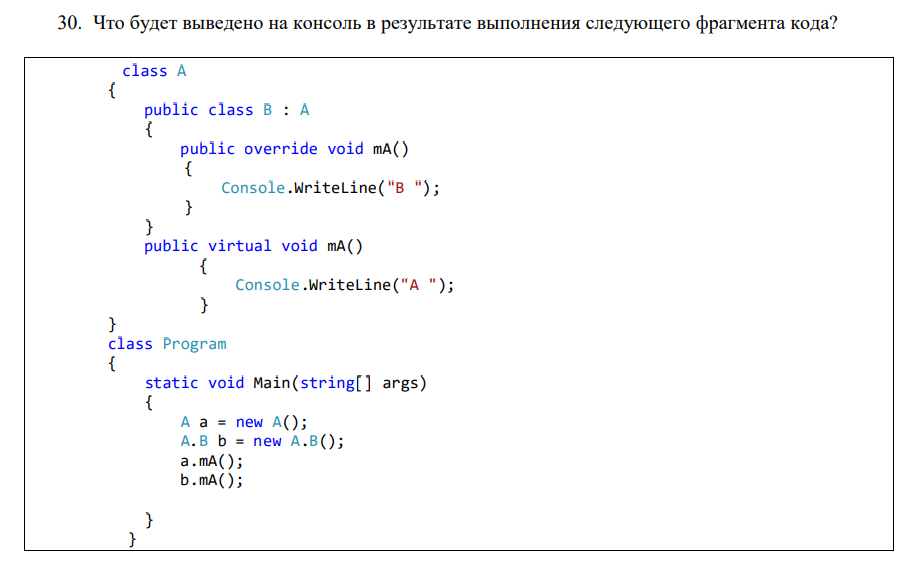
27) Для чего используются стандартные интерфейсы ICloneable, IComparable, IComparer, IEnumerable?

* ICloneable – клонирование объектов
* ICompareable – сравнение объектов для упорядочивания или сортировки своих экземпляров.
* IComparer – сравнивает два объекта.
* IEnumerable – перечислитель, который поддерживает простую итерацию по универсальной коллекции. (позволяет перебирать элементы в коллекциях)
* 



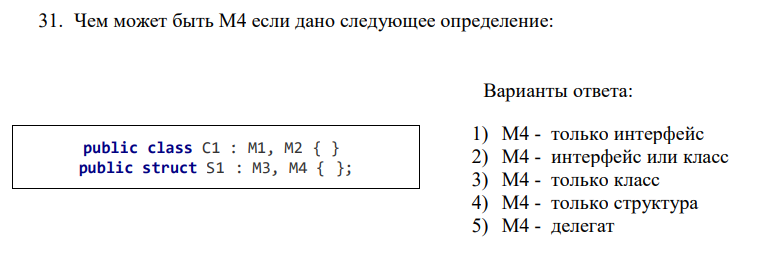


3 4



A

B



1) 2)

32) Выберите верное присваивание для объектов, определенных в листинге.

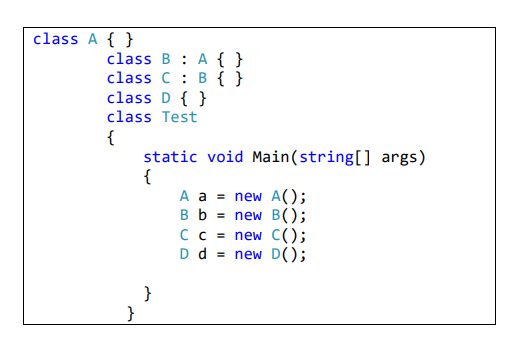
1) b = a;

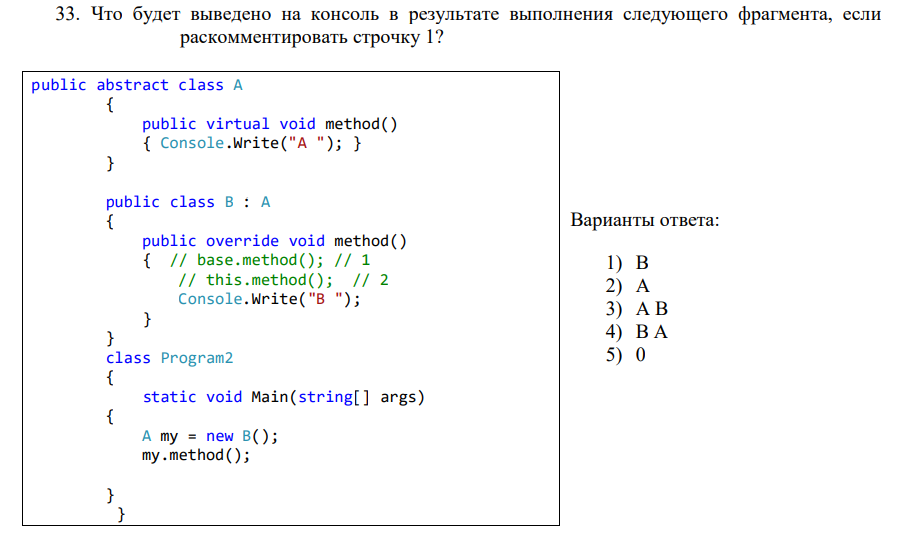
2) a = b;

3) c = a;

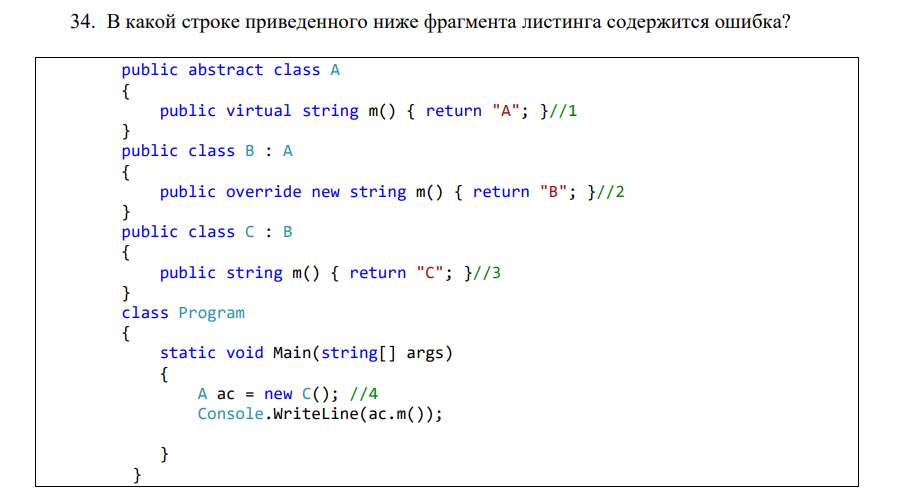
4) d = a;

5) с = b;

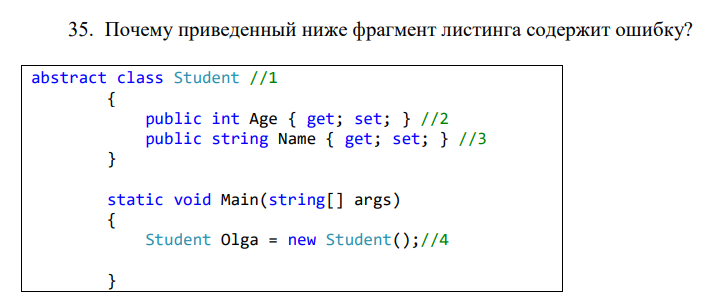




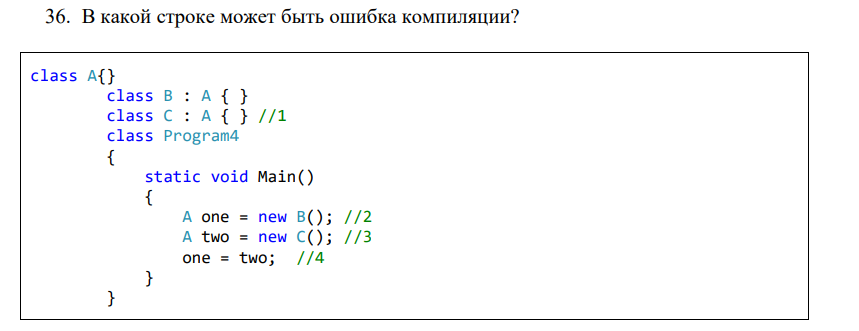
Если с комм. То B, а так 3)А B



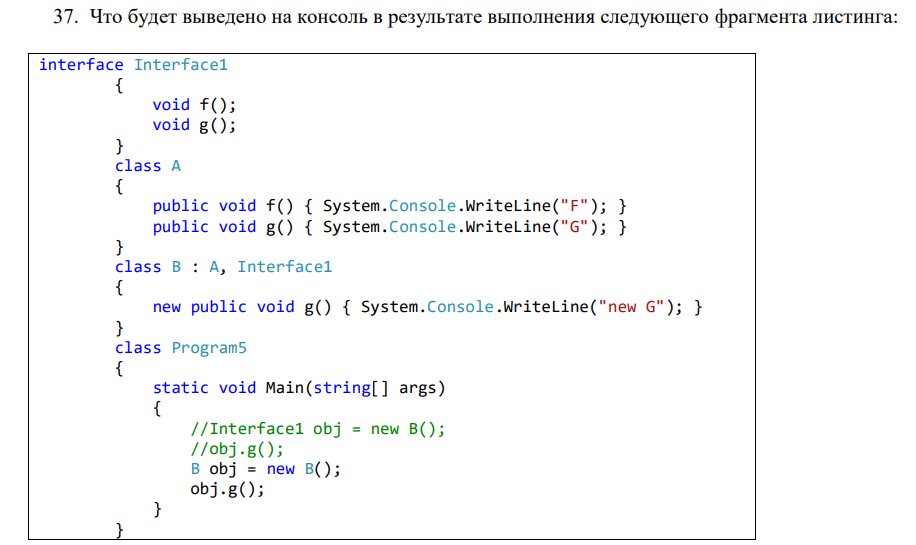
3



(4), т.к нельзя создавать объекты абстр. Класса



(4)



new G