1. Статические классы
   1. Для определения типов, которые не имеют экземпляров, а имеют только статические члены, такие как методы, свойства, поля и события.
2. Что содержит?
   1. Статические поля, общие для всех методов значения.
   2. Статические свойства – для получения и установления значений стат. Поля.
   3. Статические методы.
   4. Статические события. Позволяют оповещать другие классы о наступлении ситуации.
   5. Статический конструктор, вызывается один раз перед первым обращением к статическому классу.
3. Производный и базовый классы?
   1. Базовый – это класс, от которого происходит наследование.
   2. Производный – это класс, который основывается на базовом.
4. Base
   1. Используется для доступа к членам базового класса из производного. Base.Method(), :base() – конструктор.
5. Основная задача наследования?
   1. Позволяет создавать новые классы на основе существующих, наследуя их свойства и методы. Это способствует формированию иерархии классов. Улучшает читаемость кода, убирает дубликаты, полиморфизм.
6. BaseFunc().
   1. Может иметь доступ, если этот метод не является приватным. Нужно использовать ключевое слово base.basefunc() для вызова метода базового класса.\
7. Конструктор A и B.
   1. class B : A // класс B наследует от класса A
   2. {
   3. public B() : base() // конструктор класса B вызывает конструктор класса A
   4. {
   5. // тело конструктора класса B
   6. }
   7. }
8. Полиморфизм
   1. Это свойство объектов разных типов иметь одинаковый интерфейс и реагировать на одни и те же события разными способами. Позволяет работать с объектами, не зная их конкретного типа, а только общий базовый тип. Для реализации используются виртуальные методы и переопредлеение.
9. Назначение виртуальных функций.
   1. Это функции, которые можно переопределить в классах-наследниках, для описания из поведения в зависимости от типа объекта. Позволяют реализовать полиморфизм.
10. Protected
    1. Доступны для классов, которые наследуют от класса, а также для классов в том же пакете, что и класс, в котором они объявлены.
11. Наследуется private?
    1. Да, наследуются, но в наследниках недоступны. Производные классы не могут обращаться к ним напрямую, а только через методы базового класса.
12. As,is
    1. As – для попытки преобразования выражения к указанному типу, возвращает null, если не вышло.
    2. Is – для проверки, совместимо ли выражение с указанным типом. Int is int.
13. Множественное наследование
    1. Нельзя классы =( (но можно интерфейсы)
    2. Класс может наследовать только от одного базового класса и всех его предков.
14. Запрет наследования?
    1. Модификатор sealed.
15. Разрешить наследование, но запрет перекрытия методов?
    1. Да. Нужно использовать sealed перед объявлением метода в базовом классе.
16. Абстрактный класс?
    1. Это класс, который не может иметь экземпляров и используется в основном для создания наследников. Abstract.
    2. Абстрактный метод – это метод абстрактного класса, который не имеет реализации и должен быть переопределен в производных класссах.
17. Когда обязаны объявить абстрактный класс?
    1. Когда класс содержит хоть один абстрактный метод, т.е. метод, который не имеет реализации в базовом классе.
    2. Если есть несколько родственных объектов и им нужно придать общие функции.
18. Разница abstract и virtual?
    1. Абстрактный класс – не может быть инстанцирован. Только наследуется другими.
    2. Виртуальный класс – это класс, который, может быть, инстанцирован и наследован другими классом. Не объявляется словом virtual, а является обычным классом.
    3. Абстрактные методы не имеют реализацию и должны быть переопределены в наследнике.
    4. Виртуальный метод имеет реализацию и может быть переопределен.
19. Виртуальные компоненты.
    1. Методы, свойства, индексаторы, события,
20. Интерфейс
    1. Это набор абстрактных членов, которые могут быть реализованы классами или структурами, которые наследуют интерфейс. Для общих функций разных типов.
21. Что содержит интерфейс?
    1. Методы, свойства, индексаторы, события, статические методы, реализация по умолчанию.
22. Работа с объектом через унаследованный интерфейс?
    1. Мы можем использовать только те члены, которые объявлены в этом интерфейсе и не зависеть от конкретной реализации класса. Интерфейс IAnimal, Метод MakeSound, и два класса Cat и Dog, которые наследуют этот интерфейс и реализуют его по-разному, то мы можем создать массив объектов типа IAnimal и вызывать метод MakeSound для каждого элемента, не зная, какой животный он представляет:
23. Явная реализация интерфейса
    1. Мы указываем имя интерфейса перед именем члена класса, который реализует этот член интерфейса. Позволяет скрыть член от прямого доступа и обращаться к нему только через ссылку на интерфейс. IPrint, IDraw, Move(). => IPrint.Move() и IDraw.Move().
24. Почему нельзя модификатор доступа для методов интерфейса?
    1. Так как методы интерфейса должны быть публичными, т.к. они реализуются в других классах.
    2. Исключение – это реализация через явный интерфейс.
    3. // Определение класса, реализующего интерфейс явно
    4. public class Example : IExample
    5. {
    6. // Явная реализация метода интерфейса
    7. void IExample.DoSomething()
    8. {
    9. Console.WriteLine("Doing something explicitly");
    10. }
    11. }
    12. // Создание объекта класса Example
    13. Example example = new Example();
    14. // Вызов метода DoSomething напрямую через объект класса невозможен
    15. // example.DoSomething(); // Ошибка компиляции
    16. // Вызов метода DoSomething через ссылку на интерфейс возможен
    17. IExample iexample = example as IExample;
    18. iexample.DoSomething();
25. Интерфейсы множественное.
    1. Можно, указав интерфейсы через запятую.
26. Отличия интерфейса и абстрактного класса.
    1. Интерфейс содержит только определения методов, свойств, индексаторов и событий, но не их реализации. Абстрактный класс может содержать любые модификаторы доступа.
    2. Класс может реализовывать несколько интерфейсов, но только один абстрактный класс.
    3. Интерфейс может наследоваться от других интерфейсов, но не классов. Абстрактный класс может наследоваться от других классов и интерфейсов.
    4. Интерфейс определяет контракт или спецификацию поведения для типов, а абстракция – базу или скелет для производных типов.
27. Стандартные интерфейсы.
    1. ICloneable – для создания копии объектов.
    2. IComparable – для сравнения объектов одного типа между собой.
    3. IComparer – для сравнения объектов разных типов.
    4. IEnumerable – для перебора элементов коллекции или последовательности с помощью цикла foreach.
28. В 2 строке нет ошибки
29. 3 4
30. A B
31. 1
32. 2
33. A B
34. 2
35. 4, т.к. абстрактный класс
36. 4
37. New G

5 laba:

1. Отличие класса от структуры?
   1. Класс – ссылочный тип, а структура – значимый. Класс хранится в куче, а структура в стеке.
   2. Класс может наследовать от другого класса или интерфейса, а структура только от интерфейса.
   3. Класс имеет конструктор по умолчанию, а структура нет.
2. Что может и чего не может быть в структуре?
   1. Могут быть: поля, константы, индексаторы, свойства, методы, события, операторы, вложенные типы.
   2. Не может иметь: методы завершения, базовые классы, виртуальные или абстрактные члены. Также структура не может быть статической, абстрактной или sealed.
3. Перечисление?
   1. Это тип данных, который состоит из наборы именованных констант, связанных с числовыми значениями. Перечисление ограничено набором заданных значений.
   2. // Определить перечисление Align, которое содержит три константы: Left, Center и Right enum Align { Left, Center, Right } Left = 0, Center = 1, Right = 2., если не задано иное перечисление.
4. Стандартные интерфейсы .Net
   1. IComparable и IComparer: эти интерфейсы определяют методы для сравнения объектов по значению или по заданному критерию. Они используются для реализации операций сравнения, сортировки и поиска12.
   2. IEnumerable и IEnumerator: эти интерфейсы определяют методы для перебора элементов коллекции или последовательности.
   3. ICloneable: этот интерфейс определяет метод для создания копии объекта. Он используется для реализации клонирования объектов.
   4. IDisposable: этот интерфейс определяет метод для освобождения неуправляемых ресурсов, таких как файлы, потоки, сокеты и т.д. Он используется для реализации детерминированного финализатора78.
   5. INotifyPropertyChanged: этот интерфейс определяет событие для уведомления об изменении свойства объекта. Он используется для реализации привязки данных в WPF и других технологиях.
5. IComparable?
   1. Используется для реализации сравнения объектов определенного типа.
   2. ля использования интерфейса IComparable необходимо реализовать метод CompareTo, который принимает объект в качестве параметра и возвращает целое число, указывающее, как текущий объект относится к переданному объекту.
6. IClonable?
   1. Используется для реализации клонирования объектов
   2. Для использования интерфейса ICloneable необходимо реализовать метод Clone, который возвращает копию текущего объекта. В зависимости от того, какие поля и свойства содержит объект, можно применять разные виды клонирования: поверхностное или глубокое. Поверхностное клонирование копирует только значения примитивных типов, а ссылочные типы остаются общими для обоих объектов. Глубокое клонирование копирует все поля и свойства, включая ссылочные типы, создавая новые экземпляры для них