|  |
| --- |
| 1. Для чего используют статические классы? |
|  |  |
|  | Для использования их методов без создания экземпляра класса. Например, класс Math. |
|  |  |
|  | 2. Что может содержать статический класс? |
|  |  |
|  | Статический класс может содержать только СТАТИЧЕСКИЕ методы, свойства и поля. |
|  |  |
|  | 3. Что такое производный и базовый классы? |
|  | Производный класс (derived class) – это класс, который использует программный код базового класса и изменяет (расширяет) его под свои потребности. В других языках программирования (например, Java) базовый класс еще называется суперкласс (superclass), а производный класс называется подкласс (subclass). Базовый класс — это класс, от которого наследуются производные классы. |
|  |  |
|  | 4. 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. |
|  | Доступ осуществляется к базовому классу, заданному в объявлении класса. Например, если указать class ClassB : ClassA, члены ClassA будут доступны из ClassB независимо от базового класса ClassA. |
|  |  |
|  |  |
|  |  |
|  | 5. В чем заключена основная задача наследования? |
|  |  |
|  | Благодаря наследованию один класс может унаследовать функциональность другого класса. |
|  | производный класс может иметь доступ только к тем членам базового класса, которые определены с модификаторами private protected (если базовый и производный класс находятся в одной сборке), public, internal (если базовый и производный класс находятся в одной сборке), protected и protected internal. |
|  |  |
|  | 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, если тот не объявлен приватным. |
|  |  |
|  | 7. Напишите объявление конструктора без аргументов для производного класса B, который будет вызывать конструктор без аргументов базового класса A. |
|  |  |
|  | 8. Что такое полиморфизм? Приведите пример. |
|  | полиморфизм – это различная реализация однотипных действий. Благодаря ему можно реализовывать одинаковые действия с разными типами переменных или назначить универсальные. Осуществляется это с помощью перегрузки операторов или методов. |
|  | 9. Определите назначение виртуальных функций. |
|  | При наследовании класса данные методы м.б. переопределен в производных классах с помощью ключевого слова override. Могут быть не значит, что должны будут обязательно переодпределены, как это происходит с abstract. |
|  |  |
|  | 10. Кому доступны переменные с модификатором protected? |
|  | Такой член класса доступен из любого места в текущем классе или в производных классах. При этом производные классы могут располагаться в других сборках. |
|  |  |
|  | Любому классу-наследнику в общем. |
|  |  |
|  | 11. Наследуются ли переменные с модификатором private? |
|  | Да, но они не являются доступными. |
|  |  |
|  | 12. As, is – что это, как применяется? В чем между ними отличие ? |
|  |  |
|  | Ключевое слово IS позволяет сравнить объект и тип данных, что бы определить принадлежит ли наш объект этому типу. Возвращает данная конструкция либо true либо false. |
|  |  |
|  | Ключевое слово AS служит для перевода объекта к указанному типу, но в отличие от знакомой конструкции используемой в примере выше [(тип)объект], в случае невозможности привести объект к указанному типу мы вместо исключения получим null. |
|  | Т.е. is сранивает обьект с типом данных, а as привоит обьект к указанному типу. |
|  |  |
|  | Is Возвращает булевское значение, говорящее о том, можете ли вы преобразовать данное выражение в указанный тип. Никогда не генерирует исключение. |
|  |  |
|  | As позволяет преобразовывать тип в определенный ссылочный тип с применением следующего синтаксиса: |
|  | операнд as <тип> |
|  |  |
|  | 13. Поддерживает ли C# множественное наследование? |
|  | С# поддерживает множественное наследование в виде наследования от класса и нескольких интерфейсов, или просто от нескольких интерфейсов. |
|  | Но не поддерживает наследование от нескольких классов |
|  |  |
|  | 14. Можно ли запретить наследование от класса? |
|  | Да, с помощью модификатора sealed. |
|  |  |
|  | 15. Можно ли разрешить наследование класса, но запретить перекрытие метода? |
|  | Да. Указываем класс как public, а метод как sealed. |
|  |  |
|  | 16. Что такое абстрактный класс? |
|  | Это класс, объект которого не может быть создан. Такой класс должен иметь класс-наследник с реализацией абстрактных методов. Абстрактный класс – это фактически чертёж нормального класса без реализации. |
|  |  |
|  | 17. В каком случае вы обязаны объявить класс абстрактным? |
|  | 1. В том случае, если класс является наследником абстрактного класса, но не все методы базового класса перекрыты и имеют реализацию. |
|  | 2. В том случае, если хотя бы один метод класса является абстрактным. |
|  |  |
|  | 18. В чем разница между абстрактными и виртуальными классами? Между виртуальными и абстрактными методами? |
|  | Абстрактный класс это класс, содержащий хотя бы один метод (abstract).. |
|  | Виртуальный метод имеет реализацию и МОЖЕТ БЫТЬ переопределен в производном классе. Абстрактный метод не имеет реализацию, только описание метода, который ДОЛЖЕН БЫТЬ реализован в производных классах. |
|  |  |
|  | 19. Какие компоненты класса могут быть виртуальными? |
|  | Виртуальными могут быть: |
|  | • Методы |
|  | • Свойства |
|  | • Индексаторы |
|  | • События |
|  |  |
|  | 20. Что такое интерфейс? |
|  | Интерфейсы – это еще один инструмент реализации полиморфизма в Си-шарп. Интерфейс представляет собой набор методов (свойств, событий, индексаторов), реализацию которых должен обеспечить класс, который реализует интерфейс. |
|  |  |
|  | 21. Что может содержать 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. |
|  |  |
|  |  |
|  | 22. Как работать с объектом через унаследованный интерфейс? |
|  |  |
|  | 23. Приведите пример явной реализации интерфейса. |
|  | При явной реализации указывается название метода или свойства вместе с названием интерфейса, при этом мы не можем использовать модификатор public, то есть методы являются закрытыми: |
|  |  |
|  |  |
|  | interface IAction |
|  | { |
|  | void Move(); |
|  | } |
|  | class BaseAction : IAction |
|  | { |
|  | void IAction.Move() |
|  | { |
|  | Console.WriteLine("Move in Base Class"); |
|  | } |
|  | } |
|  | Следует учитывать, что при явной реализации интерфейса его методы и свойства не являются частью интерфейса класса. Поэтому напрямую через объект класса мы к ним обратиться не сможем: |
|  |  |
|  | static void Main(string[] args) |
|  | { |
|  | BaseAction action = new BaseAction(); |
|  | ((IAction)action).Move();   // необходимо приведение к типу IAction |
|  |  |
|  | // или так |
|  | IAction action2 = new BaseAction(); |
|  | action2.Move(); |
|  |  |
|  | Console.ReadKey(); |
|  | } |
|  |  |
|  |  |
|  | 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. |
|  |  |
|  | 26. Назовите отличия между интерфейсом и абстрактным классом. |
|  | В интерфейсе все методы (свойства и т.д.) абстрактны и не имеют реализации. В абстрактном классе некоторые методы могут быть реализованы. В интерфейсе члены не могут иметь модификатора видимости (все они являются public по умолчанию), а в абстрактном классе члены могут иметь модификатор видимости. |
|  |  |
|  | 27. Для чего используются стандартные интерфейсы ICloneable, IComparable, IComparer, lEnumerable? |
|  |  |
|  | Интерфейс IComparable задает метод сравнения объектов по принципу больше или меньше, что позволяет выполнять их сортировку. Реализация интерфейсов IEnumerable и IEnumerator дает возможность просматривать содержимое объекта с помощью конструкции foreach, а реализация интерфейса ICloneable – клонировать объекты. |
|  | Стандартные интерфейсы поддерживаются многими стандартными классами библиотеки. Например, работа с массивами с помощью цикла foreach возможна именно потому, что тип Array реализует интерфейсы IEnumerable и IEnumerator. Можно создавать и собственные классы, поддерживающие стандартные интерфейсы, что позволит использовать объекты этих классов стандартными способами |
|  |  |
|  |  |
|  | 28. В какой строке приведенного ниже фрагмента листинга не содержится ошибки? class A { public virtual abstract void m() { } //1 public virtual void g() { }//2 public virtual new new void f() { }//3 public static virtual void h() { }//4 } |
|  |  |
|  | Во второй строке. |
|  |  |
|  | 29. Что будет выведено на консоль в результате выполнения следующего фрагмента? |
|  | class A { public int x = 1; } class B : A { public new int x = 2; public void m(int a, int b) { x = a; base.x = b; Console.Write(x + " " + base.x); |
|  |  |
|  | } } class Test { static void Main(string[] args) { A a = new A(); B b = new B(); b.m(3, 4); |
|  |  |
|  | } } |
|  | Выведено будет |
|  |  |
|  | 3 4 |
|  |  |
|  | 30. Что будет выведено на консоль в результате выполнения следующего фрагмента кода? |
|  | class A { public class B : A { public override void mA() { Console.WriteLine("B "); } } public virtual void mA() { Console.WriteLine("A "); } } class Prоgrаm { static void Main(string[] args) { A a = new A(); A.B b = new A.B(); a.mA(); b.mA(); |
|  |  |
|  | } } |
|  |  |
|  | Выведено будет |
|  | А |
|  | В |
|  |  |
|  | 31. Чем может быть M4 если дано следующее определение: |
|  |  |
|  | public class C1 : M1, M2 { } |
|  | public struct S1 : M3, M4 { }; |
|  |  |
|  | Варианты ответа: |
|  | 1) M4 - только интерфейс(да) |
|  | 2) M4 - интерфейс или класс (нет) |
|  | 3) M4 - только класс (нет) |
|  | 4) M4 - только структура (нет) |
|  | 5) M4 - делегат (?) |
|  |  |
|  |  |
|  | 32. Выберите верное присваивание для объектов, определенных в листинге. |
|  | class A { } class B : A { } class C : B { } class D { } class Test { static void Main(string[] args) { A a = new A(); B b = new B(); C c = new C(); D d = new D(); |
|  |  |
|  | } } |
|  |  |
|  | Варианты ответа: |
|  | 1) b = a; |
|  | 2) a = b; |
|  | 3) c = a; |
|  | 4) d = a;(нет точно) |
|  | 5) с = b; |
|  |  |
|  | 33. Что будет выведено на консоль в результате выполнения следующего фрагмента, если раскомментировать строчку 1? |
|  | public abstract class A { public virtual void method() { Console.Write("A "); } } |
|  |  |
|  | public class B : A { public override void method() { // base.method(); // 1 // this.method(); // 2 Console.Write("B "); } } class Program2 { static void Main(string[] args) { A my = new B(); my.method(); |
|  |  |
|  | } } |
|  | Варианты ответа: |
|  | 1) B |
|  | 2) A |
|  | 3) A B (да) |
|  | 4) B A |
|  | 5) 0 |
|  |  |
|  |  |
|  | 34. В какой строке приведенного ниже фрагмента листинга содержится ошибка? |
|  | public abstract class A { public virtual string m() { return "A"; }//1 } public class B : A { public override new string m() { return "B"; }//2 } public class C : B { public string m() { return "C"; }//3 } class Prоgrаm { static void Main(string[] args) { A ac = new C(); //4 Console.WriteLine(ac.m()); |
|  |  |
|  | } } |
|  | Во второй строке. |
|  |  |
|  | 35. Почему приведенный ниже фрагмент листинга содержит ошибку? |
|  | abstract class Student //1 { public int Age { get; set; } //2 public string Name { get; set; } //3 } |
|  |  |
|  | static void Main(string[] args) { Student Olga = new Student();//4 |
|  |  |
|  | } |
|  | Потому что от абстрактного класса нельзя создавать обьекты(?) |
|  |  |
|  |  |
|  | 36. В какой строке может быть ошибка компиляции? |
|  | class A{} class B : A { } class C : A { } //1 class Program4 { static void Main() { A one = new B(); //2 A two = new C(); //3 one = two; //4 } } |
|  |  |
|  | Ошибок компиляции здесь нет. |
|  |  |
|  | 37. Что будет выведено на консоль в результате выполнения следующего фрагмента листинга: |
|  | interface Interface1 { void f(); void g(); } class A { public void f() { System.Console.WriteLine("F"); } public void g() { System.Console.WriteLine("G"); } } class B : A, Interface1 { new public void g() { System.Console.WriteLine("new G"); } } class Program5 { static void Main(string[] args) { //Interface1 obj = new B(); //obj.g(); B obj = new B(); obj.g(); } } |
|  |  |
|  | . |
|  | Выведено: |
|  | new G |