**Основные принципы ООП**

(https://training.by/News/Items/275?lang=ru)

Абстракция — отделение концепции от её экземпляра Полиморфизм — реализация задач одной и той же идеи разными способами;

Наследование — способность объекта или класса базироваться на другом объекте или классе.

Инкапсуляция — размещение одного объекта или класса внутри другого для разграничения доступа к ним.

**Методы System.Object**

https://learn.microsoft.com/ru-ru/dotnet/api/system.object.equals?view=net-6.0

**Code convention**

(https://learn.microsoft.com/ru-ru/dotnet/csharp/fundamentals/coding-style/coding-conventions)

Используйте регистр верблюда ("camelCasing") при именовании private или internal полях, а также префикс их.\_

При именовании interfaceиспользуйте регистр pascal в дополнение к префиксу имени с помощью I. Это явно указывает потребителям, что это .interface

Используйте регистр pascal ("PascalCasing") при именовании class, recordили struct.

Запись только одного оператора в строке.

Запись только одного объявления в строке.

Добавление по крайней мере одной пустой строки между определениями методов и свойств.

Использование скобок для ясности предложений в выражениях, как показано в следующем коде.

Комментарий размещается на отдельной строке, а не в конце строки кода.

Текст комментария начинается с заглавной буквы.

Текст комментария завершается точкой.

**Переопределение**

Модификатор **override.** Используется для переопределения метода.

**Интерфейс**

Интерфейс представляет некое описание типа, набор компонентов, который должен иметь тип данных.

Ключевое слово **interface**

определять следующие сущности:

Методы

Свойства

Индексаторы

События

Статические поля и константы (начиная с версии C# 8.0)

Абстрактный класс

По умолчанию доступ **public.**

**Абстрактный класс**

(https://metanit.com/sharp/tutorial/3.8.php)

ключевое слово **abstract**. (abstract class Example{})

Для описания сущностей, которые не имеют конкретного воплощения, предназначены абстрактные классы. Абстрактный класс похож на обычный класс. Отличия:

Мы **НЕ можем** использовать конструктор абстрактного класса для создания экземпляра класса

**Виртуальные методы**

(https://metanit.com/sharp/tutorial/3.19.php)

При наследовании нередко возникает необходимость изменить в классе-наследнике функционал метода, который был унаследован от базового класса.

модификатор **virtual** (public virtual void Function(){})

Чтобы переопределить метод в классе-наследнике, этот метод определяется с модификатором **override**

**Ключевое слово base**

Кроме конструкторов, мы можем обратиться с помощью ключевого слова **base** к другим членам базового класса

class Employee : Person{

    public string Company { get; set; }

public Employee(string name, string company):base(name)

{ Company = company;}

    public override void Print(){

        base.Print();

        Console.WriteLine($"работает в {Company}");

    }

}

**Класс типа sealed**

Запретить переопределение методов и свойств: модификатором **sealed**

class Employee : Person{

    public string Company { get; set; }

public Employee(string name, string company):base(name)

{ Company = company;}

    public override sealed void Print(){

        base.Print();

        Console.WriteLine($"{Name} работает в {Company}");

    }

}

**Одноимённые методы - перегрузка**

(https://metanit.com/sharp/tutorial/3.5.php)

**Ссылки на абстрактные классы и интерфейсы**

https://professorweb.ru/my/csharp/charp\_theory/level9/9\_2.php

**Операторы is или as**

<https://professorweb.ru/my/csharp/charp_theory/level9/9_2.php>

https://www.bestprog.net/ru/2019/03/09/run-time-type-information-statements-is-as-examples-ru/#:~:text=%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%20is%20%E2%80%93%20%D0%BF%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D1%8F%D0%B5%D1%82%2C%20%D1%81%D0%BE%D0%B2%D0%BF%D0%B0%D0%B4%D0%B0%D0%B5%D1%82%20%D0%BB%D0%B8,%D0%BE%20%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%BD%D0%BE%D0%BC%20%D1%82%D0%B8%D0%BF%D0%B5%20(%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B5).

оператор is – проверяет, совпадает ли тип выражения с заданным типом данных;

оператор as – предназначен для избежания возникновения исключительной ситуации в случае неудачного приведения типов;

поддерживает ли данный тип тот или иной интерфейс

**типе объекта и его текущих значениях**

Название класса

1. Для чего используют статические классы?

Для определения статических полей. Используется один раз.

1. Что может содержать статический класс?

Содержит только статические члены.

1. Что такое производный и базовый классы?

Класс, члены которого наследуются, называется базовым классом. Класс, который наследует члены базового класса, называется производным классом.

Производный класс наследует описание базового класса

1. Как используют ключевое слово base?

base. или : base()

1. В чем заключена основная задача наследования?

Если есть абстрактный класс фигура, передать все методы, свойства конкретным классам.

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, если basefunc() наследуется

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

Class B(){

A a = new A();

}

1. Что такое полиморфизм? Приведите пример.

Функции обрабатывать данные разных типов.

Функция складывающая double числа может складывать int числа.

1. Определите назначение виртуальных функций.

связывание вызова функции с ее определением выполняется на этапе выполнения программы.

Методы и свойства, которые мы хотим сделать доступными для переопределения, в базовом классе помечается модификатором virtual.

1. Кому доступны переменные с модификатором protected?

доступен из любого места в своем классе или в производных классах

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 ?

оператор is – проверяет, совпадает ли тип выражения с заданным типом данных;

оператор as – приведение к типу

If(obj is SomeType);

1. Поддерживает ли C# множественное наследование?

в C# не поддерживается множественное наследование. Поддерживает только для интерфейсов.

1. Можно ли запретить наследование от класса?

Да, можно, при помощи модификатора sealed

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

Да

1. Что такое абстрактный класс?

Класс который наследуется, но по его образцу нельзя создавать объекты.

1. В каком случае вы обязаны объявить класс абстрактным?

если класс является наследником абстрактного класса, но не все методы базового класса перекрыты и имеют реализацию

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

абстрактные методы используются для описания методов, которые должны иметь все производные классы, виртуальный же метод предоставляет возможность переопределить метод в производном классе

1. Какие компоненты класса могут быть виртуальными?

Методы

Свойства

Индексаторы

События

1. Что такое интерфейс?

граница между двумя функциональными объектами, требования к которой определяются стандартом; совокупность средств, методов и правил взаимодействия между элементами системы.

Грубо говоря требования к классу.

1. Что может содержать интерфейс?

Интерфейсы могут содержать методы экземпляра, свойства, события, индексаторы, а также любое сочетание этих четырех типов членов. Интерфейсы могут содержать статические конструкторы, поля, константы или операторы. Начиная с C# 11, элементы интерфейса, которые не являются полями, могут быть static abstract .

1. Как работать с объектом через унаследованный интерфейс?

Interface obj = SomeObject;

1. Приведите пример явной реализации интерфейса.

interface IAction

{

void Move();

}

class BaseAction : IAction

{

void IAction.Move() => Console.WriteLine("Move in Base Class");

}

1. Почему нельзя указать модификатор видимости для методов интерфейса?

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

1. Можно ли наследовать от нескольких интерфейсов?

Да

1. Назовите отличия между интерфейсом и абстрактным классом.

интерфейс определяет только поведение

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

https://studopedia.su/10\_146239\_standartnie-interfeysi-NET.html

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.

28. В какой строке приведённого 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

}

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);

}

}

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();

}

}

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

2) M4 - интерфейс или класс

3) M4 - только класс

4) M4 - только структура

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;

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

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());

}

}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

}

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

}

}

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();

}

}