1. Назовите принципы ООП. Поясните каждый из них.

Абстракция — это принцип, который позволяет выделять важные характеристики объекта, скрывая при этом несущественные детали.

Инкапсуляция — это принцип, который объединяет данные (состояние объекта) и методы (функции объекта), которые работают с этими данными, в одном классе, и скрывает реализацию данных от внешнего вмешательства.

Наследование — это возможность создавать новые классы на основе существующих, унаследуя их свойства и методы.

Полиморфизм — это возможность объектов разных классов реагировать на одно и то же сообщение (вызов метода) по-разному. Полиморфизм позволяет одному интерфейсу работать с разными типами объектов.

Композиция

1. Назовите класс .NET, от которого наследуются все классы. **System.Object**
2. Охарактеризуйте открытые методы System.Object.

 **ToString()** — возвращает строковое представление объекта. По умолчанию — имя типа, можно переопределить для пользовательских объектов.

 **Equals(object obj)** — проверяет равенство текущего объекта с другим. По умолчанию сравнивает ссылки на объекты, но может быть переопределён для сравнения содержимого.

 **GetHashCode()** — возвращает хэш-код объекта, используется в структурах данных, таких как хэш-таблицы. Можно переопределить для уникальности на основе содержимого.

 **GetType()** — возвращает объект Type, представляющий тип текущего экземпляра. Используется для получения информации о типе объекта во время выполнения.

1. Охарактеризуйте закрытые методы System.Object.

 **Finalize()** — вызывается сборщиком мусора для очистки ресурсов перед удалением объекта.

 **MemberwiseClone()** — создает поверхностную копию объекта (копирует поля, но не глубокие ссылки).

1. Приведите пример определения класса.

6. class, public, private, protected, internal, protected internal, static, abstract, sealed, partial, readonly, const

7. В чем отличие между объектом и классом?

 Класс — это абстрактное описание сущности.

 Объект — это конкретная реализация этого описания.

8. Что такое конструктор? Когда вызывается конструктор?

**Конструктор** — это специальный метод класса, автоматически вызываемый при создании нового объекта этого класса. Его основная задача — инициализация объекта. Конструктор имеет такое же имя, как и класс, и не имеет возвращаемого типа. Он вызывается при использовании оператора new для создания экземпляра класса.

Вот свойства конструктора:

9. Перечислите свойства конструктора?

\*\*Имя конструктора\*\*: Совпадает с именем класса.

\*\*Нет возвращаемого типа\*\*: Конструктор не имеет возвращаемого типа, даже `void`.

\*\*Автоматический вызов\*\*: Конструктор вызывается автоматически при создании объекта класса.

\*\*Инициализация\*\*: Используется для установки начальных значений полей объекта и выполнения других начальных операций.

\*\*Перегрузка\*\*: Может быть перегружен, то есть один класс может иметь несколько конструкторов с разными параметрами.

\*\*Параметры\*\*: Может принимать параметры, которые используются для задания начальных значений свойств или полей объекта.

10. Что такое деструктор (destructor) ?

Деструктор (destructor) — это специальный метод в объектно-ориентированном программировании, который автоматически вызывается при уничтожении объекта. Он используется для выполнения операций по очистке ресурсов, таких как освобождение памяти, закрытие файлов или соединений с базами данных

11. Что такое this?

this — это ключевое слово в объектно-ориентированных языках программирования, которое ссылается на текущий экземпляр класса.

### 13. Спецификаторы доступа в C#

- \*\*`public`\*\*, \*\*`private`\*\*, \*\*`protected`\*\*, \*\*`internal`\*\*, \*\*`protected internal`\*\*, \*\*`private protected`\*\*.

### 14. Модификатор `protected internal`

- Член доступен в пределах текущего сборки или производных классов.

### 15. Параметры `ref` и `out`

- \*\*`ref`\*\*: передает параметр по ссылке, инициализация параметра обязательна.

- \*\*`out`\*\*: передает параметр по ссылке, но инициализация до передачи не требуется.

### 16. Пример необязательных и именованных параметров

```csharp

void PrintInfo(string name, int age = 18)

{

Console.WriteLine($"{name}, {age}");

}

PrintInfo(age: 30, name: "John"); // Имя и возраст указаны явно

```

### 17. Пример полей класса

```csharp

class Example

{

public static int StaticField = 10;

public const int ConstantField = 100;

public readonly int ReadonlyField = 20;

public Example(int value)

{

ReadonlyField = value;

}

}

```

### 18. Пример свойств класса

```csharp

class MyClass

{

private int \_age;

public int Age

{

get { return \_age; }

set { \_age = value > 0 ? value : 0; }

}

}

```

Свойства предоставляют механизм для инкапсуляции данных, контролируя доступ к полям класса.

### 19. Имя параметра метода `set`

- Явное имя параметра — \*\*`value`\*\*.

### 20. Автоматические свойства

- Это свойства, для которых компилятор автоматически создает закрытое поле.

```csharp

public int MyProperty { get; set; }

```

### 21. Индексаторы класса

- Позволяют классу работать как массив.

```csharp

public string this[int index] { get { return array[index]; } set { array[index] = value; } }

```

\*\*Ограничение\*\*: индексатор не может быть статическим.

### 22. Перегруженный метод

- Это метод с тем же именем, но разными параметрами.

### 23. Partial класс

- Позволяет разделить определение класса на несколько файлов. Это удобно для разделения кода и работы с большими проектами.

### 24. Анонимный тип

- Тип без имени, создается на основе значений:

```csharp

var person = new { Name = "John", Age = 30 };

```

### 25. Статические классы

- Используются для хранения методов и свойств, которые не зависят от экземпляра класса.

### 26. Статическое поле vs Экземплярное поле

- \*\*Статическое поле\*\* принадлежит классу и общее для всех его объектов.

- \*\*Экземплярное поле\*\* — уникально для каждого объекта.

### 27. Работа статических конструкторов

- Вызываются один раз, перед первым использованием статических членов класса.

### 28. Поверхностное и глубокое копирование

- \*\*Поверхностное копирование\*\* копирует только ссылки на объекты.

- \*\*Глубокое копирование\*\* создает полную копию объекта, включая все вложенные объекты.

### 29. Разница между равенством и тождеством объектов

- \*\*Равенство (`Equals`)\*\*: сравнение значений объектов.

- \*\*Тождество (`==`)\*\*: сравнение ссылок на объекты.

### 30. Частичные классы и методы

- Позволяют разделять реализацию класса на несколько файлов. Частичные методы могут быть определены, но не обязательно реализованы.