 **Принципы ООП:**

* **Абстракция:** Представление только необходимых характеристик объекта, скрытие сложных деталей.
* **Наследование:** Возможность создания новых классов на основе существующих, с заимствованием их свойств и методов.
* **Инкапсуляция:** Скрытие данных и предоставление к ним доступа только через определенные методы (использование модификаторов доступа).
* **Полиморфизм:** Возможность использования одного и того же имени метода с разными реализациями.

 **Класс .NET, от которого наследуются все классы:**  
Все классы в .NET наследуются от базового класса System.Object.

 **Открытые методы System.Object:**

* **Equals:** Сравнивает текущий объект с другим объектом.
* **GetHashCode:** Возвращает хэш-код объекта.
* **ToString:** Возвращает строковое представление объекта.
* **GetType:** Возвращает тип текущего экземпляра.

 **Закрытые методы System.Object:**

* **Finalize:** Очищает ресурсы перед сборкой мусора. Это аналог деструктора в C#.
* **MemberwiseClone:** Создает поверхностную копию текущего объекта.

 **Пример определения класса:**

csharp

Копировать код

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

public void SayHello()

{

Console.WriteLine($"Hello, my name is {Name}.");

}

}

 **Ключевые слова при определении класса:**

* public, private, protected, internal, abstract, sealed, static, partial.

 **Отличие между объектом и классом:**

* **Класс:** это шаблон или схема, описывающая поведение и состояние объектов.
* **Объект:** это экземпляр класса, созданный по его шаблону.

 **Конструктор:**  
Метод, который инициализирует объект при его создании. Он вызывается при создании объекта с помощью ключевого слова new.

 **Свойства конструктора:**

* Имеет такое же имя, как и класс.
* Не имеет возвращаемого значения.
* Может быть перегружен (с разными параметрами).

 **Деструктор (destructor):**  
Метод, который вызывается перед удалением объекта для освобождения ресурсов. В C# это метод с тильдой (~) перед именем класса.

 **this:**  
Ссылка на текущий экземпляр класса. Используется для доступа к полям и методам объекта.

 **Что будет выведено:**

csharp

Копировать код

5 5

7 7

Объекты a и b указывают на один и тот же экземпляр класса.

 **Спецификаторы доступа:**

* public, private, protected, internal, protected internal, private protected.

 **protected internal:**  
Доступен внутри текущей сборки и из классов-наследников в других сборках.

 **ref и out:**

* ref: Передает параметр по ссылке, позволяет изменять значение аргумента.
* out: Передает параметр по ссылке, но его нужно обязательно инициализировать в методе.

 **Пример необязательных и именованных параметров:**

csharp

Копировать код

void PrintMessage(string message = "Hello", string name = "World")

{

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

}

PrintMessage(name: "Alice"); // Выведет: Hello, Alice!

 **Пример полей:**

* Статическое поле: public static int counter;
* Константное поле: public const int MaxValue = 100;
* Поле только для чтения: public readonly int id;

 **Пример свойств класса:**

csharp

Копировать код

public class Car

{

public string Model { get; set; }

public int Year { get; private set; }

}

Свойства обеспечивают инкапсуляцию, позволяя контролировать доступ к полям.

 **Явное имя параметра set:**  
Имя параметра по умолчанию — value.

 **Автоматические свойства:**  
Свойства, которые не требуют явного объявления поля. Пример:

csharp

Копировать код

public string Name { get; set; }

 **Индексаторы класса:**  
Позволяют обращаться к объектам класса как к массивам:

csharp

Копировать код

public double this[int index]

{

get { return stack[index]; }

}

Ограничение: индексаторы не могут быть статическими.

 **Перегруженный метод:**  
Метод, который имеет одинаковое имя, но разное количество или тип параметров.

 **Partial класс:**  
Разделение определения класса на несколько файлов. Преимущества — удобство работы с большими классами.

 **Анонимный тип в C#:**  
Тип, который создается на лету и имеет свойства, но не требует явного определения класса.

 **Статические классы:**  
Используются для создания классов, которые содержат только статические члены и не могут быть инстанциированы.

 **Статическое поле vs экземплярное поле:**

* **Статическое поле:** принадлежит классу, общее для всех экземпляров.
* **Экземплярное поле:** принадлежит конкретному экземпляру класса.

 **Статические конструкторы:**  
Инициализируют статические поля класса. Вызываются один раз перед использованием класса.

 **Поверхностное vs глубокое копирование:**

* **Поверхностное копирование:** копируются только ссылки на объекты.
* **Глубокое копирование:** копируются сами объекты и их поля.

 **Равенство vs тождество объектов:**

* **Равенство:** объекты могут иметь одинаковое значение, но быть разными экземплярами.
* **Тождество:** два объекта — это один и тот же экземпляр.

 **Частичные классы и методы:**

* **Частичный класс:** класс может быть определен в нескольких файлах.
* **Частичный метод:** метод может быть определен частично в одном файле и реализован в другом.