ТЕОРИЯ

1) **Чем класс отличается от структуры?**

* **Класс** — это ссылочный тип данных, который создается в куче (heap). Экземпляры классов передаются по ссылке. Классы могут содержать методы, свойства, конструкторы, деструкторы и другие элементы. Наследование, полиморфизм и инкапсуляция активно используются с классами.
* **Структура** — это значимый тип данных, который создается в стеке (stack). Экземпляры структур передаются по значению. Структуры предназначены для небольших объектов, которым не требуется сложное управление памятью. Структуры не поддерживают наследование (кроме реализации интерфейсов) и не могут иметь деструкторы.

2)**Что может и чего не может быть в структуре?**

* **Может быть:**
  + Поля, свойства, методы.
  + Конструкторы (включая конструкторы с параметрами).
  + Статические члены.
  + Реализация интерфейсов.
* **Не может быть:**
  + Переопределение методов.
  + Наследование от других классов или структур (структуры наследуются от ValueType).
  + Явные инициализаторы для полей (их нужно инициализировать в конструкторе).
  + Абстрактные и виртуальные методы.
  + нельзя объявить деструктор (метод завершения) в типе структуры

3) **Что такое перечисление? Приведите пример определения и использования перечисления.** Перечисление (enum) — это набор именованных целочисленных констант, которые определяют возможные значения переменной. Это позволяет улучшить читабельность и удобство кода.

**Пример определения:**

enum DayOfWeek

{

Monday,

Tuesday,

Wednesday,

Thursday,

Friday,

Saturday,

Sunday

}

**Пример использования:**

DayOfWeek today = DayOfWeek.Tuesday;

if (today == DayOfWeek.Tuesday)

{

Console.WriteLine("Сегодня вторник.");

}

4) **Перечислите и поясните стандартные интерфейсы .Net:**

* **IComparable** — используется для сравнения объектов одного типа для сортировки или упорядочения.
* **IEnumerable** — предоставляет механизм для перебора коллекции элементов.
* **ICollection** — расширяет интерфейс IEnumerable и поддерживает базовые методы для управления коллекциями (например, добавление и удаление элементов).
* **IDisposable** — предоставляет метод для освобождения ресурсов, используемых объектом.
* **ICloneable** — предоставляет метод для создания копии объекта.

5)**Как используется интерфейс IComparable?** Интерфейс IComparable используется для создания объектов, которые можно сравнивать друг с другом (например, для сортировки). Этот интерфейс определяет метод CompareTo, который возвращает:

* Отрицательное число, если объект меньше переданного.
* Ноль, если они равны.
* Положительное число, если объект больше переданного.

**Пример:**

class Person : IComparable<Person>

{

public string Name { get; set; }

public int Age { get; set; }

public int CompareTo(Person other)

{

return this.Age.CompareTo(other.Age);

}

}

6) **Как используется интерфейс ICloneable?** Интерфейс ICloneable предоставляет метод Clone(), который создает копию объекта. Он может возвращать как глубокую, так и поверхностную копию объекта, в зависимости от реализации.

**Пример:**

class Person : ICloneable

{

public string Name { get; set; }

public int Age { get; set; }

public object Clone()

{

return this.MemberwiseClone(); // поверхностное копирование

}

}

7) **Что такое полиморфизм? Перечислите его формы. Приведите примеры.** Полиморфизм — это способность объектов разных типов предоставлять одно и то же поведение через один и тот же интерфейс.

* способность к изменению функций, унаследованных от базового класса

**Формы полиморфизма:**

* **Компиляторный (перегрузка)** — одно имя метода, но разные параметры.
* **Раннее связывание (статический полиморфизм)** — перегрузка методов и операторов.
* **Позднее связывание (динамический полиморфизм)** — использование виртуальных методов и интерфейсов.

**Пример перегрузки:**

class Calculator

{

public int Add(int a, int b) => a + b;

public double Add(double a, double b) => a + b;

}

**Пример динамического полиморфизма:**

class Animal

{

public virtual void Speak()

{

Console.WriteLine("Animal sound");

}

}

class Dog : Animal

{

public override void Speak()

{

Console.WriteLine("Bark");

}

}

Animal myDog = new Dog();

myDog.Speak(); // Вывод: Bark

8)) **Зачем в классе определяют виртуальные методы?** Виртуальные методы используются для того, чтобы позволить подклассам изменять (переопределять) поведение методов базового класса. Это является основой для динамического полиморфизма.

**Пример:**

class Animal

{

public virtual void MakeSound()

{

Console.WriteLine("Some animal sound");

}

}

class Dog : Animal

{

public override void MakeSound()

{

Console.WriteLine("Bark");

}

}

9) **Как сделать запрет переопределения методов?** Чтобы запретить переопределение метода в наследуемых классах, его нужно объявить как sealed. Это запрещает дальнейшую модификацию метода в подклассах.

**Пример:**

class Animal

{

public virtual void MakeSound()

{

Console.WriteLine("Some sound");

}

}

class Dog : Animal

{

public sealed override void MakeSound()

{

Console.WriteLine("Bark");

}

}

class Bulldog : Dog

{

// Ошибка: нельзя переопределить метод MakeSound, так как он помечен как sealed

}