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

Структура в C# является типом передающимся по значению, класс же является ссылочным типом.

В структуре создается объект значение, но не объект ссылки, структура не может наследоваться от классов, но может от интерфейса.

1. **Что может и чего не может быть в структуре?**

Структуру можно создавать без new, не может объявляться конструктор без параметров и не может инициализировать поля если они не const и static

1. **Что такое перечисление? Приведите пример определения и использования перечисления**

Перечисление-именованный проиндексированный список литералов

enum Winter\_Month

{ December, January, Febrary }

1. **Перечислите и поясните стандартные интерфейсы .Net?**

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.

1. **Как используется интерфейс IComparable?**

Интерфейс IComparable содержит всего один метод **CompareTo(),**возвращающий результат сравнения двух объектов – текущего и переданного ему в качеству параметра.

1. **Как используется интерфейс ICloneable?**

**Клонирование**-создание копии объекта. Копия объекта называется **клоном.** При присваивании одного экземпляра другому копируется ссылка, а не сам объект. Если необходимо скопировать в другую область памяти поля объекта, можно воспользоваться методом **MemberwiseClone(),** который любой объект наследует от класса object. При этом объекты, на которые указывают поля объекта, в свою очередь являющиеся ссылками, не копируются. Это называется ***поверхностным клонированием***. Для создания полностью независимых объектов необходимо ***глубокое копирование****,* когда в памяти создается дубликат всего дерева объектов, то есть объектов, на которые ссылаются поля объекта, поля полей, и т.д. Алгоритм глубокого копирования сложен, требует рекурсивного обхода всех ссылок объекта и отслеживания циклических зависимостей.

Объект, имеющий собственные алгоритмы клонирования, должен объявляться как производный интерфейса ICloneable и переопределять его единственный метод Clone().

Модернизируем предыдущий пример и создадим поверхностную копию экземпляра класса Monster с помощью метода MemberwiseClone() и реализации интерфейса ICloneable. Метод MemberwiseClone() можно вызывать только из методов класса, так как он объявлен в классе object как protected.

1. **Что такое полиморфизм? Перечислите его формы. Приведите примеры.**

Полиморфизм – это взаимозаменяемость объектов с одинаковым интерфейсом. Полиморфизм – это один из столпов ооп, заключается он во фразе "один шаблон, множество реализаций" как пример, переопределение методов, свойств, полей, конструкторов

Методы и свойства? которые мы хотим переопределить помечаются ключевым словом **virtual.** Они представляют полиморфный интерфейс. При переопределении в классе- наследнике используется ключевое слово **override**

Class person

{

public virtual void Writing (string str)

{

Console.WriteLine (str + “this is overrided method”);

}

}

Class Employee : Person

{

public override void Writing (string str)

{

Console.WriteLine (str + “this is overrided method”);

}

}

1. **Зачем в классе определяют виртуальные методы?**

Для их дальнейшего переопределения.

1. **Как сделать запрет переопределения методов?**

Указатель для модификатора sealed