1) - Описать класс Animal с полями Age и Name;
   - Создать несколько элементов Animal и положить их в список ArrayList;
   - Отбразить элементы из списка на консоль через foreach;
   - Попробовать отсортировать элементы через вызов Sort класса List; Выползет исключение, после которого надо объяснить - почему оно выползло;
   - Наследовать Animal от IComparable и реализовать метод CompareTo(Object);
   - Проверить, как работает сортировка путем повторного отображения элементов списка на консоль;
   - Изменить список ArrayList на List<> - проверить, что все по прежнему работает, если не работает - доработать;
   - Наследовать Animal от IComparable<> и реализовать метод CompareTo(Animal);

2) - Сделать класс графа MyGraph, хранящий внутри себя граф типа дерево, состоящий из узлов типа Node.
   - Реализовать возможность использования экземпляра класса в цикле foreach через итераторы для обхода в глубину (три способа) и в ширину.
   - Реализация через метод TreeTraversal, в который передается тип обхода TreeTraversalType.

```
MyGraph g = new MyGraph();
```

```
...
... (заполнение графа)
...

foreach(Node n in
g.TreeTraversal(TreeTraversalType.Left))
   Console.WriteLine(n)
```

3 (не обязательно, кто сможет!) - Есть некая программа и у нее есть классы, наследованные от абстрактного класса Plugin (Plugin1 и Plugin2)
с методами: String Description() и String DoWork();
- Есть окно с элементами: ListBox, TextBox и Button;
- В элемент ListBox помещаются результаты вызова метода Description;
- При нажатии на кнопку, производится вызов метода DoWork выбранного экземпляра из элемента ListBox,
результат вызова метода помещается в элемент TextBox;
- Необходимо сделать поддержку заранее неизвестных нашей программе типов с контрактом Plugin (т.е. сделать интерфейс IPlugin и реализовать программу, чтобы она зависела от этого интерфейса, а не от класса Plugin),
пока просто добавив пару классов, независимых от иерархии Plugin;
- Реализовать возможность динамической подгрузки типов и их инстанциирования в момент

запуска программы:
сделать это через анализ списка DLL в папке программы;

4) Сделать массив двумя способами:
ArrayList и List<>
Проверить скорость вставки и получения элемента для случая хранения типов int и string.
Используйте миллион или 10 миллионов операций.
Сделать проверку в виде generic-метода с параметрами: тип хранилища и тип проверки

5) Есть интерфейс IToolKit с методом string[] GetTools()
Есть интерфейс IParts с методом string[] GetParts()
Есть классы мебели не generic!, которые наследуются от вышеприведенных интерфейсов и хранят внутри себя список инструментов и составных элементов (Стул, Стол и т.п.)

Создать generic класс FurnitureKit<T>, T - может быть одним из классов мебели и дополнять его названием, цветом и методом для вывода на экран списка инструментов и составных частей.
public abstract class IkeaKit<TContents> where TContents : IToolKit, IParts, new()