# Интерфейсы

1. Создайте интерфейс `IStorage<T>`, который будет определять базовые операции для хранения и получения данных типа `T` в хранилище.

2. Определите в интерфейсе методы:
   - `void Add(T item)`: добавляет элемент `item` в хранилище.
   - `T Get(int index)`: возвращает элемент из хранилища по указанному индексу.
   - `int Count { get; }`: свойство, возвращающее общее количество элементов в хранилище.

3. Создайте класс `ListStorage<T>`, который реализует интерфейс `IStorage<T>`. В этом классе используйте обобщенную коллекцию `List<T>` для хранения элементов.

4. Реализуйте методы интерфейса `IStorage<T>` в классе `ListStorage<T>`:
   - Добавление элемента в хранилище должно быть реализовано с помощью метода `Add()` коллекции `List<T>`.
   - Получение элемента из хранилища по индексу должно быть реализовано с помощью индексатора коллекции `List<T>`.
   - Количество элементов в хранилище должно быть реализовано с помощью свойства `Count` коллекции `List<T>`.

5. Создайте класс `Program` с методом `Main()`, где вы можете создать объект класса `ListStorage<T>`, добавить элементы в хранилище и получить элементы из хранилища.

## IClonable

1. Создайте класс "Person" с полями "Name" и "Age". Реализуйте интерфейс IClonable для этого класса, используя глубокое копирование.

2. Создайте класс "Rectangle" с полями "Width" и "Height". Реализуйте интерфейс IClonable для этого класса, используя поверхностное копирование.

## IComparable и IComparer

1. Создайте класс "Book" с полями "Title" и "Author". Реализуйте интерфейс IComparable для этого класса, чтобы книги можно было сравнивать по названию.

2. Создайте класс "Car" с полями "Make" и "Model". Реализуйте интерфейс IComparer для этого класса, чтобы можно было сравнивать машины по марке и модели. В задании требуется реализовать несколько вариантов сравнения.

## IEnumerable и IEnumerator

1. Создайте класс "MyList" с полем "Items" типа List<int>. Реализуйте интерфейс IEnumerable для этого класса, чтобы можно было перебирать элементы списка.

2. Создайте класс "MyDictionary" с полем "Items" типа Dictionary<string, int>. Реализуйте интерфейс IEnumerator для этого класса, чтобы можно было перебирать элементы словаря.

3. Создайте метод "GetEvenNumbers", который будет возвращать все четные числа из заданного диапазона. Используйте итератор и команду yield для реализации этого метода.

## Индексаторы

Создайте класс "Matrix" с полем "Data" типа двумерного массива. Реализуйте индексатор для этого класса, чтобы можно было получать и устанавливать значения в матрице по индексам.

## Методы расширения

1. Создайте метод расширения "IsPalindrome", который будет проверять, является ли строка палиндромом.

2. Создайте метод расширения "ToTitleCase", который будет преобразовывать первую букву каждого слова в строке в верхний регистр.