Практическое занятие 1.

Создать и реализовать интерфейсы, также использовать наследование и полиморфизм для указанных предметных областей. В классе должны быть реализованы:
✓ поля (закрытые или защищённые), нужное количество;
✓ свойства (для каждого поля);
✓ конструкторы (не менее двух);
✓ перегруженные операции;
✓ методы (нужное количество);
✓ метод ToString();
✓ статические поля, если они нужны;
✓ статические свойства (если есть статические поля).

В 11 варианте необходимо реализовать: interface Ткань <- abstract class Одежда <- class Костюм.

Код:

```
public interface IFabric
{
string Material { get; set; }
string Color { get; set; }
void Tear();
}

public abstract class Clothing : IFabric
{
public abstract string Material { get; set; }
public abstract string Color { get; set; }
public virtual void Tear() => Console.WriteLine($"Одежда из {Material} порвалась!");
public virtual void Wear() => Console.WriteLine($"Надеваю одежду цвета {Color} из {Material}.");
public virtual void DisplayInfo() => Console.WriteLine($"Материал: {Material}, Цвет: {Color}");
}

public class Suit : Clothing
{
private static int _totalCreated;
private string _material;
private string _color;
private string _size;
private int _buttons;

public static int TotalCreated => _totalCreated;
public static string DefaultSize { get; set; } = "M";

public Suit()
{
_size = DefaultSize;
_totalCreated++;
}
```

```
public Suit(string material, string color, string size, int buttons)
{
_material = material;
_color = color;
_size = size;
_buttons = buttons;
_totalCreated++;
}

public override string Material
{
get => _material;
set => _material = value ?? throw new ArgumentNullException(nameof(value));
}

public override string Color
{
get => _color;
set => _color = value ?? throw new ArgumentNullException(nameof(value));
}

public string Size
{
get => _size;
set => _size = !string.IsNullOrEmpty(value) ? value : DefaultSize;
}

public int Buttons
{
get => _buttons;
set => _buttons = value >= 0 ? value : 0;
}

public static bool operator ==(Suit left, Suit right)
{
if (ReferenceEquals(left, right)) return true;
if (left is null || right is null) return false;
return left._material == right._material
&& left._color == right._color
&& left._size == right._size;
}

public static bool operator !=(Suit left, Suit right) => !(left == right);

public override bool Equals(object obj) => obj is Suit other && this == other;
public override int GetHashCode() => HashCode.Combine(_material, _color, _size);

public override string ToString() =>
$"Костюм [Материал: {_material}, Цвет: {_color}, Размер: {_size}, Пуговицы: {_buttons}]";

public void AddButtons(int count) => Buttons += Math.Max(0, count);
```

```
public override void Wear() =>
Console.WriteLine($"Надеваю костюм размера {_size}. Цвет: {_color}, Пуговиц: {_buttons}");
}
```