**Курс:** Шаблоны проектирования приложений

**Тема:** Модуль 03 Базовые принципы проектирования

*Баллы:*

Произведите корректную (правильную) по вашему мнению реализацию с применением принципа DRY:

**Использование методов для устранения дублирования кода**

public class OrderService

{

public void CreateOrder(string productName, int quantity, double price)

{

double totalPrice = quantity \* price;

Console.WriteLine($"Order for {productName} created. Total: {totalPrice}");

}

public void UpdateOrder(string productName, int quantity, double price)

{

double totalPrice = quantity \* price;

Console.WriteLine($"Order for {productName} updated. New total: {totalPrice}");

}

}

**Использование общих базовых классов**

public class Car

{

public void Start()

{

Console.WriteLine("Car is starting");

}

public void Stop()

{

Console.WriteLine("Car is stopping");

}

}

public class Truck

{

public void Start()

{

Console.WriteLine("Truck is starting");

}

public void Stop()

{

Console.WriteLine("Truck is stopping");

}

}

Произведите корректную (правильную) по вашему мнению реализацию с применением принципа KISS:

**Избегание чрезмерного использования абстракций**

public interface IOperation

{

void Execute();

}

public class AdditionOperation : IOperation

{

private int \_a;

private int \_b;

public AdditionOperation(int a, int b)

{

\_a = a;

\_b = b;

}

public void Execute()

{

Console.WriteLine(\_a + \_b);

}

}

public class Calculator

{

public void PerformOperation(IOperation operation)

{

operation.Execute();

}

}

**Избегание избыточного использования шаблонов проектирования**

public class Singleton

{

private static Singleton \_instance;

private Singleton() { }

public static Singleton Instance

{

get

{

if (\_instance == null)

{

\_instance = new Singleton();

}

return \_instance;

}

}

public void DoSomething()

{

Console.WriteLine("Doing something...");

}

}

public class Client

{

public void Execute()

{

Singleton.Instance.DoSomething();

}

}

Произведите корректную (правильную) по вашему мнению реализацию с применением принципа YAGNI:

**Избыточное создание абстракций**

public interface IShape

{

double CalculateArea();

}

public class Circle : IShape

{

private double \_radius;

public Circle(double radius)

{

\_radius = radius;

}

public double CalculateArea()

{

return Math.PI \* \_radius \* \_radius;

}

}

public class Square : IShape

{

private double \_side;

public Square(double side)

{

\_side = side;

}

public double CalculateArea()

{

return \_side \* \_side;

}

}

**Излишняя параметризация методов**

public class MathOperations

{

public int Add(int a, int b, bool shouldLog = false)

{

int result = a + b;

if (shouldLog)

{

Console.WriteLine($"Result: {result}");

}

return result;

}

}