**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;

System.out.println("Order for " + productName + " created. Total: " + totalPrice);

}

public void updateOrder(String productName, int quantity, double price) {

double totalPrice = quantity \* price;

System.out.println("Order for " + productName + " updated. New total: " + totalPrice);

}

}

**Использование общих базовых классов**

public class Car {

public void start() {

System.out.println("Car is starting");

}

public void stop() {

System.out.println("Car is stopping");

}

}

public class Truck {

public void start() {

System.out.println("Truck is starting");

}

public void stop() {

System.out.println("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 implements IOperation {

private int a;

private int b;

public AdditionOperation(int a, int b) {

this.a = a;

this.b = b;

}

@Override

public void execute() {

System.out.println(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 getInstance() {

if (instance == null) {

instance = new Singleton();

}

return instance;

}

public void doSomething() {

System.out.println("Doing something...");

}

}

public class Client {

public void execute() {

Singleton.getInstance().doSomething();

}

}

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

**Избыточное создание абстракций**

public interface IShape {

double calculateArea();

}

public class Circle implements IShape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

public class Square implements IShape {

private double side;

public Square(double side) {

this.side = side;

}

@Override

public double calculateArea() {

return side \* side;

}

}

**Излишняя параметризация методов**

public class MathOperations {

public int add(int a, int b, boolean shouldLog) {

int result = a + b;

if (shouldLog) {

System.out.println("Result: " + result);

}

return result;

}

}