Лабораторная работа № 1
Разработка программ с использованием принципа единственной обязанности (SRP) и принципа открытости/закрытости (OCP)

Цель работы: получить навыки разработки программ с использованием принципа единственной обязанности (SRP) и принципа открытости/закрытости (OCP).

1. Теоретические сведения

Принцип единственной обязанности

Принцип единственной обязанности обозначает, что каждый объект должен иметь одну ответственность и эта ответственность должна быть полностьюинкапсулирована в класс. Все его поведения должны быть направлены исключительно на обеспечение этой ответственности.

Автор этого принципа Р. С. Мартин определяет ответственность как причину изменения и заключает, что классы должны иметь одну и только одну причину для изменений. Например, представим себе класс, который составляет и печатает отчёт. Такой класс может измениться по двум причинам:

- может измениться само содержимое отчёта;
- может измениться формат отчёта.

Логично, что оба аспекта этих причин на самом деле являются двумя разными ответственностями. SRP говорит, что в таком случае нужно разделить класс на два новых класса, для которых будет характерна только одна ответственность. Причина, почему нужно сохранять направленность классов на единственную цель в том, что это делает классы более здоровыми. Что касается класса, приведённого выше, если произошло изменение в процессе составления отчёта – есть большая вероятность, что в негодность придёт код, отвечающий за печать.

Принцип открытости/закрытости

Принцип открытости/закрытости устанавливает следующее положение: программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения.

Принцип открытости/закрытости означает, что программные сущности должны быть:

- открыты для расширения: означает, что поведение сущности может быть расширено путём создания новых типов сущностей;
- закрыты для изменения: в результате расширения поведения сущности, не должны вноситься изменения в исходный код этой сущности и код, который эту сущность использует.

Таким образом, последующие изменения должны быть реализованы с помощью добавления нового кода, а не изменения уже существующего.

Нарушение принципа открытости-закрытости приводит к ситуациям, когда изменение в одном модуле вынуждает менять другие, связанные с ним.

Основным инструментом для реализации принципа открытости/закрытости является использование абстракции базовых классов.

2. Задания к лабораторной работе

Для заданного варианта задания разработать UML-диаграмму классов и диаграмму последовательности.

Разработать программу решения задания в виде консольного приложения (C++, C#) с использованием принципа единственной обязанности (SRP) и принципа открытости/закрытости (OCP). Допускается вводить дополнительные понятия предметной области. В программе предусмотрите тестирование функциональности созданных объектов классов.

Отчет по лабораторной работе – файл формата pdf. Формат имени файла отчета: <НомерГруппы >_<ФамилияСтудента>.pdf. В отчет включить построенные диаграммы и исходный код программы (при необходимости и заголовочные файлы). Формат отчета см. в Приложении. Отчет загрузить в LMS.

При защите лабораторной работы: уметь объяснить логику и детали работы программы; реализацию принципов из раздела 1 на примере разработанной программы.

Варианты заданий

1. Система Факультатив. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.
2. Система Платежи. Клиент имеет Счет в банке и Кредитную Карту (КК). Клиент может оплатить Заказ, сделать платеж на другой Счет, заблокировать КК и аннулировать Счет. Администратор может заблокировать КК за превышение кредита.
3. Система Больница. Пациенту назначается лечащий Врач. Врач может сделать назначение Пациенту (процедуры, лекарства, операции). Медсестра или другой Врач выполняют назначение. Пациент может быть выписан из Больницы по окончании лечения, при нарушении режима или при иных обстоятельствах.

Распределение вариантов заданий

№ по списку группы	№ варианта	№ по списку группы	№ варианта	№ по списку группы	№ варианта
1	1	11	2	21	3
2	2	12	3	22	1
3	3	13	1	23	2
4	1	14	2	24	3
5	2	15	3	25	1
6	3	16	1	26	2
7	1	17	2	27	3
8	2	18	3	28	1
9	3	19	1	29	2
10	1	20	2	30	3

Технологии конструирования программного обеспечения

Отчет по лабораторной работе № 00

Группа: 000-000 **Студент**: Иванов Иван Иванович

Задание на лабораторную работу

<Формулировка задания согласно варианту
……
……..>

Диаграмма классов

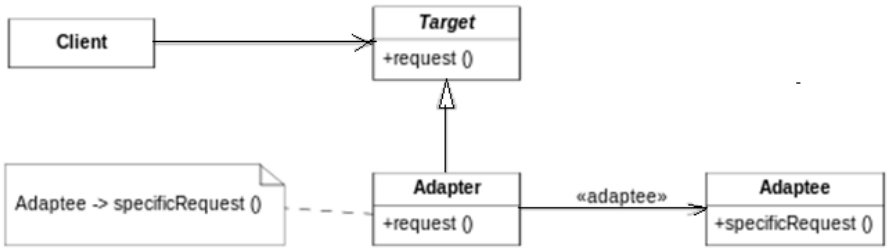
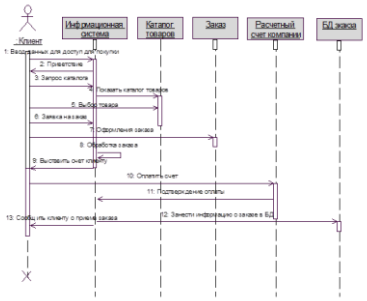


Диаграмма последовательности



Исходный код программы

```
#include "pch.h"
#include "CppUnitTest.h"
#include "../Add/Add.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1
{
	TEST_CLASS(UnitTest1)
	{
	public:

		TEST_METHOD(TestMethod1)
		{
			Assert::AreEqual(2, add(1, 1));
		}
```

……