Занятие № 19

Номер учебной группы: П-16

Фамилия, инициалы учащегося: Украинец М.И.

Дата выполнения работы: 30.11.2022

Тема работы: «Использование шаблонов при разработке программного продукта»

**Задание 1**

Изучил теоретический материал по теме «Шаблоны проектирования».

**Задание 2**

Разработал шаблоны проектирования и реализации для будущего использования при разработке ПО.

**Шаблон проектирования** (реализация на С++)

/\*\*

\* Целевой класс объявляет интерфейс, с которым может работать клиентский код.

\*/

class Target {

public:

virtual ~Target() = default;

virtual std::string Request() const {

return "Target: The default target's behavior.";

}

};

/\*\*

\* Адаптируемый класс содержит некоторое полезное поведение, но его интерфейс

\* несовместим с существующим клиентским кодом. Адаптируемый класс нуждается в

\* некоторой доработке, прежде чем клиентский код сможет его использовать.

\*/

class Adaptee {

public:

std::string SpecificRequest() const {

return ".eetpadA eht fo roivaheb laicepS";

}

};

/\*\*

\* Адаптер делает интерфейс Адаптируемого класса совместимым с целевым

\* интерфейсом с помощью множественного наследования.

\*/

class Adapter : public Target, public Adaptee {

public:

Adapter() {}

std::string Request() const override {

std::string to\_reverse = SpecificRequest();

std::reverse(to\_reverse.begin(), to\_reverse.end());

return "Adapter: (TRANSLATED) " + to\_reverse;

}

};

/\*\*

\* Клиентский код поддерживает все классы, использующие целевой интерфейс.

\*/

void ClientCode(const Target \*target) {

std::cout << target->Request();

}

int main() {

std::cout << "Client: I can work just fine with the Target objects:\n";

Target \*target = new Target;

ClientCode(target);

std::cout << "\n\n";

Adaptee \*adaptee = new Adaptee;

std::cout << "Client: The Adaptee class has a weird interface. See, I don't understand it:\n";

std::cout << "Adaptee: " << adaptee->SpecificRequest();

std::cout << "\n\n";

std::cout << "Client: But I can work with it via the Adapter:\n";

Adapter \*adapter = new Adapter;

ClientCode(adapter);

std::cout << "\n";

delete target;

delete adaptee;

delete adapter;

return 0;

}

**Шаблон реализации** (реализация на С++)

/\*\*

\* Паттерн Итератор

\*

\* Назначение: Даёт возможность последовательно обходить элементы составных

\* объектов, не раскрывая их внутреннего представления.

\*/

#include <iostream>

#include <string>

#include <vector>

/\*\*

\* C++ has its own implementation of iterator that works with a different

\* generics containers defined by the standard library.

\*/

template <typename T, typename U>

class Iterator {

public:

typedef typename std::vector<T>::iterator iter\_type;

Iterator(U \*p\_data, bool reverse = false) : m\_p\_data\_(p\_data) {

m\_it\_ = m\_p\_data\_->m\_data\_.begin();

}

void First() {

m\_it\_ = m\_p\_data\_->m\_data\_.begin();

}

void Next() {

m\_it\_++;

}

bool IsDone() {

return (m\_it\_ == m\_p\_data\_->m\_data\_.end());

}

iter\_type Current() {

return m\_it\_;

}

private:

U \*m\_p\_data\_;

iter\_type m\_it\_;

};

/\*\*

\* Конкретные Коллекции предоставляют один или несколько методов для получения

\* новых экземпляров итератора, совместимых с классом коллекции.

\*/

template <class T>

class Container {

friend class Iterator<T, Container>;

public:

void Add(T a) {

m\_data\_.push\_back(a);

}

Iterator<T, Container> \*CreateIterator() {

return new Iterator<T, Container>(this);

}

private:

std::vector<T> m\_data\_;

};

class Data {

public:

Data(int a = 0) : m\_data\_(a) {}

void set\_data(int a) {

m\_data\_ = a;

}

int data() {

return m\_data\_;

}

private:

int m\_data\_;

};

void ClientCode() {

std::cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Iterator with int\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << std::endl;

Container<int> cont;

for (int i = 0; i < 10; i++) {

cont.Add(i);

}

Iterator<int, Container<int>> \*it = cont.CreateIterator();

for (it->First(); !it->IsDone(); it->Next()) {

std::cout << \*it->Current() << std::endl;

}

Container<Data> cont2;

Data a(100), b(1000), c(10000);

cont2.Add(a);

cont2.Add(b);

cont2.Add(c);

std::cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Iterator with custom Class\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << std::endl;

Iterator<Data, Container<Data>> \*it2 = cont2.CreateIterator();

for (it2->First(); !it2->IsDone(); it2->Next()) {

std::cout << it2->Current()->data() << std::endl;

}

delete it;

delete it2;

}

int main() {

ClientCode();

return 0;

}

**Задание 3**

Ответил на контрольные вопросы.

1. Что вы понимаете под шаблонами проектирования?

Шаблон проектирования или паттерн в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

1. Перечислите плюсы и минусы применения шаблонов.

**Плюсы:** это скорость, которой можно допиться за счет либо использования огромной базы готовых шаблонов для всевозможных CMS, либо за счет наличие специализированных IDE, где вполне возможно даже за один день «состряпать» весьма приличный шаблон. Для новичка – это вообще рай на земле: не нужно осваивать ни HTML, ни CSS, ни PHP – требования (в плане освоения навыков) минимальны.

**Минусы:** фронтенда: скорость – это конечно же хорошо, равно как и минимальные вложения средств, но в итоге большинство сайтов мало чем отличаются (визуальная часть) друг от друга…, в сущности типичный, скучный и однообразный дизайн + похожая навигация.

1. Перечислите типы шаблонов проектирования и реализации.

[Шаблон делегирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%B4%D0%B5%D0%BB%D0%B5%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), [шаблон функционального дизайна](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD%D0%B0), [интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [интерфейс-маркер](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81-%D0%BC%D0%B0%D1%80%D0%BA%D0%B5%D1%80_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [канал событий](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB_%D1%81%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)).

1. Дайте характеристику шаблону проектирования «Делегирование»

Шаблон делегирование — [основной](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F&action=edit&redlink=1) [шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), в котором [объект](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) внешне выражает некоторое [поведение](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5), но в реальности передаёт ответственность за выполнение этого поведения связанному объекту.

1. Дайте характеристику шаблону проектирования «Функциональный дизайн».

Функциональный дизайн ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Functional design) — [основной](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F&action=edit&redlink=1) [шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), использующийся для упрощения проектирования [ПО](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

1. Дайте характеристику шаблону проектирования «Интерфейс».

Интерфейс ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) interface) — основной [шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), являющийся общим [методом](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) для структурирования [компьютерных программ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) для того, чтобы их было проще понять.

1. Дайте характеристику шаблону проектирования «Абстрактная фабрика».

Абстрактная фабрика ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Abstract factory) — [порождающий шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%BE%D0%B6%D0%B4%D0%B0%D1%8E%D1%89%D0%B8%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), предоставляет интерфейс для создания семейств взаимосвязанных или взаимозависимых объектов, не специфицируя их конкретных классов.

1. Дайте характеристику шаблону проектирования «Строитель».

Строитель ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Builder) — [порождающий шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%BE%D0%B6%D0%B4%D0%B0%D1%8E%D1%89%D0%B8%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) предоставляет способ создания составного объекта.

1. Дайте характеристику шаблону проектирования «Фабричный метод».

Фабричный метод ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Factory Method), или виртуальный конструктор ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Virtual Constructor) — [порождающий шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%BE%D0%B6%D0%B4%D0%B0%D1%8E%D1%89%D0%B8%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), предоставляющий [подклассам (дочерним классам)](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) [интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) для создания экземпляров некоторого класса.

1. Дайте характеристику шаблону проектирования «Прототип».

Прототип, ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Prototype) — [порождающий шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%BE%D0%B6%D0%B4%D0%B0%D1%8E%D1%89%D0%B8%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Задаёт виды создаваемых объектов с помощью экземпляра-прототипа и создаёт новые объекты путём копирования этого прототипа.

11. Дайте характеристику шаблону проектирования «Наблюдатель».

Наблюдатель ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Observer) — [поведенческий](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [шаблон проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Реализует у класса механизм, который позволяет объекту этого класса получать оповещения об изменении состояния других объектов и тем самым наблюдать за ними.