**Занятие № 19**

**Номер учебной группы:** П-16.

**Фамилия, инициалы учащегося:** Пачко Н.Н.

**Дата выполнения работы:** 30.11.2022.

**Тема работы:** «Использование шаблонов при разработке программного продукта»

**Ход работы**

**Задание 1**

Изучить теоретический материал по теме «Шаблоны проектирования».

Шаблон проектирования или паттерн (англ. design pattern) в разработке программного обеспечения — повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются идиомами. Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют архитектурные шаблоны, они охватывают собой архитектуру всей программной системы. Алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления, так как решают вычислительные задачи.

**Плюсы**

В сравнении с полностью самостоятельным проектированием, шаблоны обладают рядом преимуществ. Основная польза от использования шаблонов состоит в снижении сложности разработки за счёт готовых абстракций для решения целого класса проблем. Шаблон даёт решению своё имя, что облегчает коммуникацию между разработчиками, позволяя ссылаться на известные шаблоны. Таким образом, за счёт шаблонов производится унификация деталей решений: модулей, элементов проекта, — снижается количество ошибок. Применение шаблонов концептуально сродни использованию готовых библиотек кода. Правильно сформулированный шаблон проектирования позволяет, отыскав удачное решение, пользоваться им снова и снова. Набор шаблонов помогает разработчику выбрать возможный, наиболее подходящий вариант проектирования.

**Минусы**

Многие шаблоны проектирования в объектно-ориентированном проектировании можно рассматривать как идиоматическое воспроизведение элементов функциональных языков. Питер Норвиг утверждает, что 16 из 23 шаблонов, описанных в книге «Банды Четверых», в Lisp или Dylan реализуются существенно проще, чем в С++, либо оказываются незаметны Пол Грэхэм считает саму идею шаблонов проектирования — антипаттерном, сигналом о том, что система не обладает достаточным уровнем абстракции, и необходима её тщательная переработка. Нетрудно видеть, что само определение шаблона как «готового решения, но не прямого обращения к библиотеке» по сути означает отказ от повторного использования в пользу дублирования. Это, очевидно, может быть неизбежным для сложных систем при использовании языков, не поддерживающих комбинаторы и полиморфизм типов, и это в принципе может быть исключено в языках, обладающих свойством гомо знаковости (хотя и не обязательно эффективно), так как любой шаблон может быть реализован в виде исполнимого кода.

**Задание 2**

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

**Шаблон проектирования**

**// Общий интерфейс посредников.**

**interface Mediator is**

**method notify(sender: Component, event: string)**

**class AuthenticationDialog implements Mediator is**

**private field title: string**

**private field loginOrRegisterChkBx: Checkbox**

**private field loginUsername, loginPassword: Textbox**

**private field registrationUsername, registrationPassword,**

**registrationEmail: Textbox**

**private field okBtn, cancelBtn: Button**

**constructor AuthenticationDialog() is**

**method notify(sender, event) is**

**if (sender == loginOrRegisterChkBx and event == "check")**

**if (loginOrRegisterChkBx.checked)**

**title = "Log in"**

**// 1. Показать компоненты формы входа.**

**// 2. Скрыть компоненты формы регистрации.**

**else**

**title = "Register"**

**// 1. Показать компоненты формы регистрации.**

**// 2. Скрыть компоненты формы входа.**

**if (sender == okBtn && event == "click")**

**if (loginOrRegister.checked)**

**// Попробовать найти пользователя с данными из**

**// формы логина.**

**if (!found)**

**// Показать ошибку над формой логина.**

**else**

**// 1. Создать пользовательский аккаунт с данными**

**// из формы регистрации.**

**// 2. Авторизировать этого пользователя.**

**class Component is**

**field dialog: Mediator**

**constructor Component(dialog) is**

**this.dialog = dialog**

**method click() is**

**dialog.notify(this, "click")**

**method keypress() is**

**dialog.notify(this, "keypress")**

**// Конкретные компоненты не связаны между собой напрямую. У них**

**// есть только один канал общения — через отправку уведомлений**

**// посреднику.**

**class Button extends Component is**

**// ...**

**class Textbox extends Component is**

**// ...**

**class Checkbox extends Component is**

**method check() is**

**dialog.notify(this, "check")**

**// ...**

**Шаблон реализации**

**class** **GameAI** **is**

**method** turn() **is**

collectResources()

buildStructures()

buildUnits()

attack()

**method** collectResources() **is**

**foreach** (s in **this**.builtStructures) do

s.collect()

**abstract** **method** buildStructures()

**abstract** **method** buildUnits()

**method** attack() **is**

enemy = closestEnemy()

**if** (enemy == **null**)

sendScouts(map.center)

**else**

sendWarriors(enemy.position)

**abstract** **method** sendScouts(position)

**abstract** **method** sendWarriors(position)

**class** **OrcsAI** **extends** GameAI **is**

**method** buildStructures() **is**

**if** (there are some resources) **then**

// Строить фермы, затем бараки, а потом цитадель.

**method** buildUnits() **is**

**if** (there are plenty of resources) **then**

**if** (there are no scouts)

// Построить раба и добавить в группу

// разведчиков.

**else**

**method** sendScouts(position) **is**

**if** (scouts.length > 0) **then**

// Отправить разведчиков на позицию.

**method** sendWarriors(position) **is**

**if** (warriors.length > 5) **then**

**class** **MonstersAI** **extends** GameAI **is**

**method** collectResources() **is**

// Ничего не делать.

**method** buildStructures() **is**

// Ничего не делать.

**method** buildUnits() **is**

// Ничего не делать.

**Задание 3**

Ответьте на контрольные вопросы.

**1. Что вы понимаете** **под шаблонами проектирования?**

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

**2. Перечислите плюсы и минусы применения шаблонов.**

**Ответ:** 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.

**3. Перечислите типы шаблонов проектирования и реализации.**

**Ответ:** Шаблон делегирования, шаблон функционального дизайна, интерфейс, интерфейс-маркер, канал событий.

**4. Дайте характеристику шаблону проектирования «Делегирование»**

**Ответ:** Шаблон делегирование — основной шаблон проектирования, в котором объект внешне выражает некоторое поведение, но в реальности передаёт ответственность за выполнение этого поведения связанному объекту.

**5. Дайте характеристику шаблону проектирования «Функциональный дизайн».**

**Ответ:** Функциональный дизайн (англ. Functional design) — основной шаблон проектирования, использующийся для упрощения проектирования ПО.

**6. Дайте характеристику шаблону проектирования «Интерфейс».**

**Ответ:** Интерфейс (англ. interface) — основной шаблон проектирования, являющийся общим методом для структурирования компьютерных программ для того, чтобы их было проще понять.

**7. Дайте характеристику шаблону проектирования «Абстрактная фабрика».**

**Ответ:** Абстрактная фабрика (англ. Abstract factory) — порождающий шаблон проектирования, предоставляет интерфейс для создания семейств взаимосвязанных или взаимозависимых объектов, не специфицируя их конкретных классов.

**8. Дайте характеристику шаблону проектирования «Строитель».**

**Ответ:** Строитель (англ. Builder) — порождающий шаблон проектирования предоставляет способ создания составного объекта.

**9. Дайте характеристику шаблону проектирования «Фабричный метод».**

**Ответ:** Фабричный метод (англ. Factory Method), или виртуальный конструктор (англ. Virtual Constructor) — порождающий шаблон проектирования, предоставляющий подклассам (дочерним классам) интерфейс для создания экземпляров некоторого класса.

**10. Дайте характеристику шаблону проектирования «Прототип».**

**Ответ:** Прототип, (англ. Prototype) — порождающий шаблон проектирования. Задаёт виды создаваемых объектов с помощью экземпляра-прототипа и создаёт новые объекты путём копирования этого прототипа.

**11. Дайте характеристику шаблону проектирования «Наблюдатель».** Наблюдатель (англ. Observer) — поведенческий шаблон проектирования. Реализует у класса механизм, который позволяет объекту этого класса получать оповещения об изменении состояния других объектов и тем самым наблюдать за ними.