Учреждения образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра экономической информатики

**Отчёт по лабораторной работе №6**

«Паттерны»

Проверила: Выполнил:

Волошко Е.А. Козлов А.А.

Минск 2013

1. **Паттерны**

Шаблон проектирования или паттерн 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.

Паттерны подразделяются на:

* Порождающие;
* Поведенческие;
* Структурные.

Порождающие шаблоны (Creational) – шаблоны проектирования, которые абстрагируют процесс инстанцирования. Они позволяют сделать систему независимой от способа создания, композиции и представления объектов. Шаблон, порождающий классы, использует наследование, чтобы изменять инстанцируемый класс, а шаблон, порождающий объекты, делегирует инстанцирование другому объекту. К порождающим паттернам относятся: абстрактная фабрика, строитель, одиночка, отложенная инициализация.

Структурные шаблоны (Structural) определяют различные сложные структуры, которые изменяют интерфейс уже существующих объектов или его реализацию, позволяя облегчить разработку и оптимизировать программу. К структурным паттернам относятся: заместитель, адаптер, мост, компоновщик.

Поведенческие шаблоны (Behavioral) определяют взаимодействие между объектами, увеличивая таким образом его гибкость. К поведенческим шаблонам проектирования относятся: команда, итератор, цепочка ответственности, хранитель.

**Порождающие шаблоны:**

Абстрактная фабрика (англ. Abstract factory) – порождающий шаблон проектирования, позволяющий изменять поведение системы, варьируя создаваемыми объектами, при этом сохраняя интерфейсы. Он позволяет создавать целые группы взаимосвязанных объектов, которые, будучи созданными одной фабрикой, реализуют общее поведение. Шаблон реализуется созданием абстрактного класса Factory, который представляет собой интерфейс для создания компонентов системы (например, для оконного интерфейса он может создавать окна и кнопки). Затем пишутся наследники от него классы, реализующие этот интерфейс.

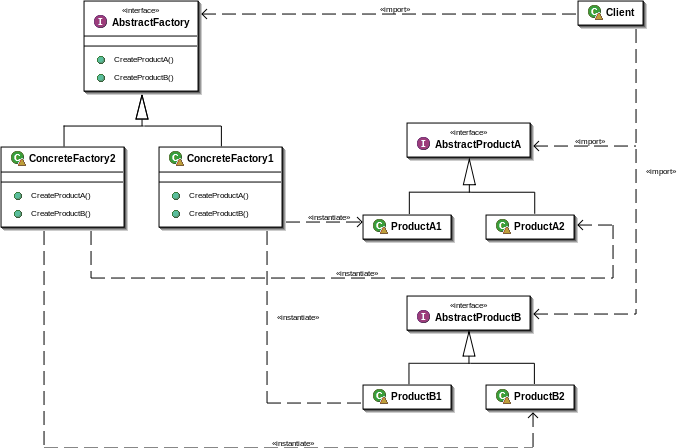


Рисунок 1 – Диаграмма классов паттерна Абстрактная фабрика

Одиночка (англ. Singleton) – порождающий шаблон проектирования, гарантирующий, что в однопоточном приложении будет единственный экземпляр класса с глобальной точкой доступа.

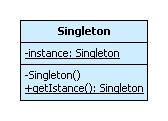


Рисунок 2 – Диаграмма классов паттерна Одиночка

Строитель (англ. Builder) – порождающий шаблон проектирования. Отделяет конструирование сложного объекта от его представления, так что в результате одного и того же процесса конструирования могут получаться разные представления.

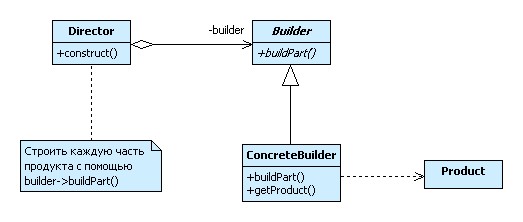


Рисунок 3 – Диаграмма классов паттерна Строитель

Адаптер (англ. Adapter или англ. Wrapper – Обёртка) – структурный шаблон проектирования, предназначенный для организации использования функций объекта, недоступного для модификации, через специально созданный интерфейс.

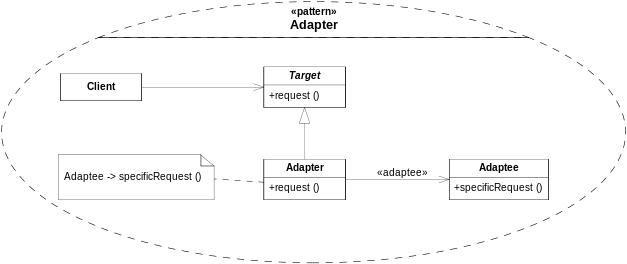


Рисунок 4 – Диаграмма классов паттерна Адаптер

Заместитель (англ. Proxy) – структурный шаблон проектирования, который предоставляет объект, который контролирует доступ к другому объекту, перехватывая все вызовы (выполняет функцию контейнера).

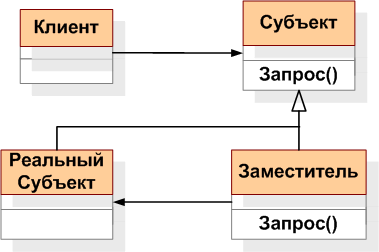


Рисунок 5 – Диаграмма классов паттерна Заместитель

Посетитель (англ. Visitor) – поведенческий шаблон проектирования, описывающий операцию, которая выполняется над объектами других классов. При изменении Visitor нет необходимости изменять обслуживаемые классы.

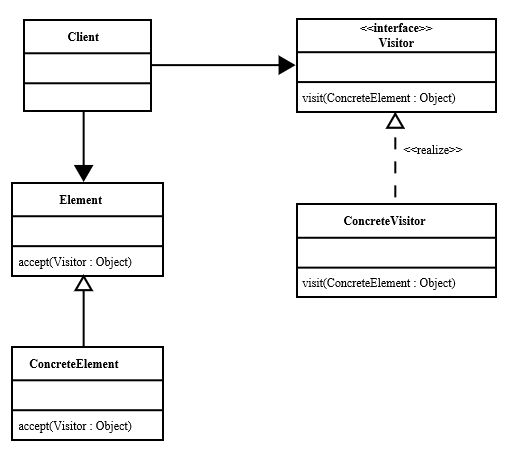


Рисунок 6 – Диаграмма классов паттерна Посетитель

Команда (англ. Command) — поведенческий шаблон проектирования, используемый при объектно-ориентированном программировании, представляющий действие. Объект команды заключает в себе само действие и его параметры.

Листинг кода с примером реализации паттерна Команда.

package commandkozlovaa;

public interface Command {

public void execute();

}

package commandkozlovaa;

public class GarageDoor {

public void up() {

System.out.println("Door is up!");

}

public void down() {

System.out.println("Door is down!");

}

public void stop() {

System.out.println("Door is stop!");

}

public void lightOn() {

System.out.println("Light is on!");

}

public void lightOff() {

System.out.println("Light is off!");

}

}

package commandkozlovaa;

public class GarageDoorOpenCommand implements Command{

GarageDoor garageDoor;

public void execute() {

garageDoor.up();

garageDoor.lightOn();

}

public GarageDoorOpenCommand( GarageDoor garageDoor) {

this.garageDoor = garageDoor;

}

}

package commandkozlovaa;

public class Light {

public void on() {

System.out.println("Light is on!");

};

public void off() {

System.out.println("Light is off!");

};

}

package commandkozlovaa;

public class LightOnCommand implements Command{

Light light;

public void execute() {

light.on();

}

public LightOnCommand(Light light) {

this.light = light;

}

}

package commandkozlovaa;

public class RemoteControl {

Command[] onCommands;

Command[] offCommands;

}

package commandkozlovaa;

public class RemoteControlTest {

public static void main(String[] args) {

SimpleRemoteControl remote =

new SimpleRemoteControl();

Light light = new Light();

GarageDoor garageDoor = new GarageDoor();

LightOnCommand lightOn = new LightOnCommand(light);

GarageDoorOpenCommand garageOpen =

new GarageDoorOpenCommand(garageDoor);

remote.setCommand(lightOn);

remote.buttonWasPressed();

remote.setCommand(garageOpen);

remote.buttonWasPressed();

}

}

package commandkozlovaa;

public class SimpleRemoteControl {

Command slot;

public SimpleRemoteControl() {}

public void setCommand(Command command) {

slot = command;

}

public void buttonWasPressed() {

slot.execute();

}

}



Рисунок 7 – Диаграмма классов паттерна Команда

1. **Subversion (SVN)**

Subversion — это бесплатная система управления версиями с открытым исходным кодом. Subversion позволяет управлять файлами и каталогами, а так же сделанными в них изменениями во времени. Это позволяет восстановить более ранние версии данных, даёт возможность изучить историю всех изменений. Благодаря этому многие считают систему управления версиями своего рода «машиной времени».

Subversion может работать через сеть, что позволяет использовать её на разных компьютерах. В какой то степени, возможность большого количества людей не зависимо от их местоположения совместно работать над единым комплектом данных поощряет сотрудничество. Когда нет того ответственного звена цепи, того контролирующего элемента, который утверждает все изменения, работа становится более эффективной. При этом не нужно опасаться, что отказ от контролирующего элемента повлияет на качество, ведь благодаря сохранению истории изменений, даже если при изменении данных будут допущены ошибки, всегда можно сделать откат изменений к прежнему состоянию.

Некоторые системы управления версиями выступают также в качестве систем управления конфигурацией программного обеспечения (SCM). Такие системы специально созданы для управления деревьями исходного кода и имеют множество особенностей, непосредственно относящихся к разработке программ: они понимают языки программирования и предоставляют инструменты для сборки программ. Subversion не является такой системой, она представляет собой систему общего назначения, которую можно использовать для управления любым набором файлов. Для Вас это будут исходники Ваших программ, а для кого-то другого это будет список продуктов или сведённое цифровое видео.