Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, механики и оптики

Факультет инфокоммуникационных технологий

Практическая работа 4

«Проектирование инфокоммуникационных систем»

Выполнил: студент группы № К3420 Бейлин Максим Тимурович

Проверил: доцент ФИКТ Осипов Никита Алексеевич

Санкт-Петербург

2020

**Цель работы:**

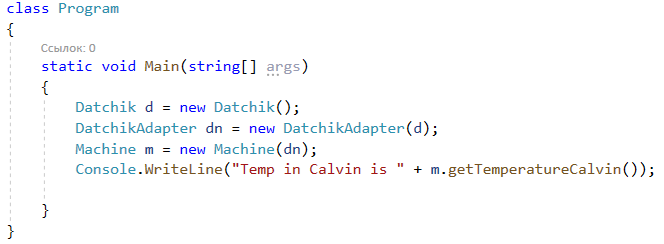
изучить шаблоны проектирования GoF и закрепить основы разработки систем на их основе.

**Задачи:**

1. Изучить шаблоны проектирования GoF [1-4], определить особенности их применения в программных системах.
2. Выполнить упражнения и контрольные задания руководства.

**Ход работы**

Первый изучаемый шаблон – **Adapter**. Суть шаблона – «адаптировать» объекты тех классов, которые не являются наследниками нужных интерфейсов, классов с целью их встраивания в общую систему и их дальнейшее использование без изменения уже имеющегося кода. Пример применения Адаптера в случае с системой климат-контроля – в готовую систему, принимающую данные в градусах Цельсия, необходимо внедрить, т. е. адаптировать датчик, выдающий данные в градусах Фаренгейта.



1. *Пример использования шаблона Адаптер*

В данном примере класс Datchik не наследуется от общего интерфейса датчиков и не может быть использован в машине Machine. Адаптер DatchikAdapter реализует нужный интерфейс и содержит в качестве свойства нестандартный датчик. А для использования нужного метода датчика в адаптере предусмотрен вызов метода датчика с обработкой данных

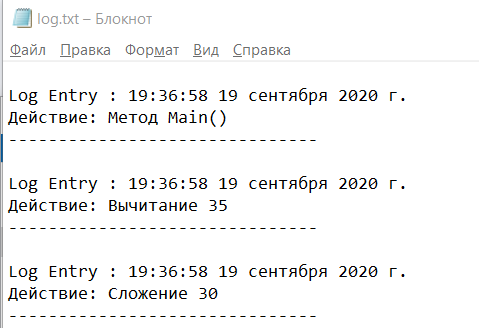
Второй изучаемый шаблон – **Abstract Factory**. Суть данного шаблона – интерфейс для создания семейств взаимосвязанных объектов с определенными интерфейсами без указания конкретных типов данных объектов.

Согласно заданию, необходимо было расширить функционал созданного приложения с уже имеющейся абстрактной фабрикой объектов-машин. Для обеспечения возможности создавать машины другой марки пришлось создавать абстрактные классы для всех соответствующих типов: для самой машины, для двигателя, для типа кузова. С одной стороны, это долгий и трудоемкий процесс. С другой стороны, это позволяет в дальнейшем не задумываться над созданием объектов по-отдельности, а позволяет с помощью одной единственной абстрактной фабрики получать всю необходимую структуру объектов (как в случае с машиной и ее составляющими), а затем работать с ней через общий интерфейс.

Третий изучаемый шаблон – **Factory Method**. Суть данного шаблона в делегировании обязанности определять класс создаваемого объекта подклассам. Конкретно в задании необходимо было помимо существующей структуры добавить новую компанию, а для нее – новый вид услуг «пьяный водитель». Смысл применяемого шаблона – это то, что на высоком уровне (там, где создается компания) не определяется непосредственно услуга, которой занимается компания. Этот процесс делегируется непосредственно на уровень компании (именно в классе компании происходит создание услуги).

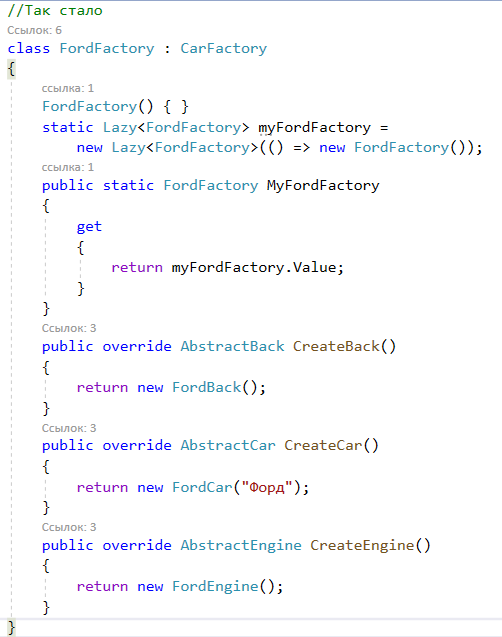
Четвертый шаблон – **Singleton**. Этот шаблон описывает классы, которые позволяют создать лишь один-единственный объект, а также содержат указания на него.

Суть задания – создать класс для логгирования действий в журнале. Задача заключается в том, чтобы объект не создавался каждый раз заново – с этой целью для класса был закрыт конструктор, объявление единственного объекта было задано в поле с использованием класса Lazy<T>, т. е. в момент обращения. Пример выполнения программы:



1. *Работа программы*

По аналогии были внесены изменения в проект AbstractFactory так, чтобы фабрика автомобилей Форд не создавалась отдельно, а использовалась как единственный объект соответствующего класса:



1. *Фабрика-одиночка*

Следующий шаблон – **Strategy**. Для выполнения разных по логике, но единых по алгоритму стратегий необходимо создать единый интерфейс-стратегию для всех вариантов обработки данных. Данный шаблон позволяет разработать несколько схожих алгоритмов и заменять их во время работы программы, т.к. работа ведется через интерфейс.

В ходе выполнения задания была добавлена стратегия сортировки пузырьком, которая реализовывала общий интерфейс. Это позволило сменить существующую стратегию без дополнительных вмешательств в код.

Аналогично реализуется и контрольное задание: для навигатора разрабатывается общий интерфейс стратегии «поиск маршрута», а затем реализуется в классах поиск и прокладку маршрута по автодорогам, пеших маршрутов, маршрутов по велодорожкам, на общественном транспорте, а также маршруты посещения достопримечательностей.

Еще один шаблон - **Template Method**. Он заключается в определении последовательности выполнения задач и их программировании в виде шаблона, т. е. абстрактного класса, содержащего вызовы для выполнения соответствующих шагов алгоритма. В задании приведен пример с вычислением прогрессий – они вычисляются по общему шаблону, но конкретная реализация зависит от типа прогрессии.

По аналогии выполняется контрольное задание. Определяется общий шаблон стрижек, а затем под мужские и женские стрижки отдельно создается алгоритм «по шаблону».

Шаблон **Façade** позволяет скрытьсложный интерфейс и свести все вызовы к одному объекту, делегирующему их соответствующим объектам системы (по сути, напоминает шаблон «Контроллер» GRASP). По заданию необходимо скрыть от пользователя сложный процесс работы микроволновой печи и организовать простой интерфейс. Объект самой печи содержит компоненты и алгоритм работы. Пользователь лишь активирует работу и видит уведомления. По аналогии с разморозкой реализуется метод жарки.

Шаблон **Цепочка обязанностей** позволяет объектам автоматически определять последовательность их работы, при этом не связывая отправителя и получателя запроса. В задании реализуется цепочка обработки денежной транзакции. Причем управление последовательностью работы обработчиков зависит как от порядка их объявления, так и от дополнительных условий обработки (в данном случае флагов). Выполнение контрольного упражнения позволило пронаблюдать работу цепочки.

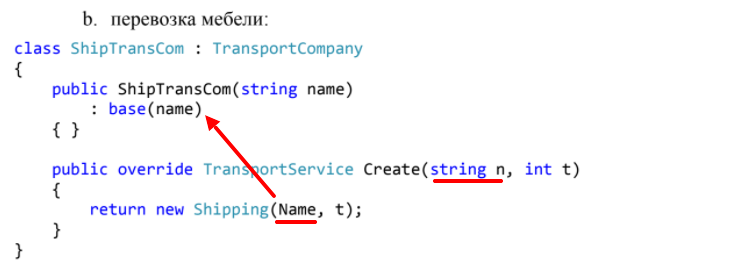
Шаблон **Command** – поведенческий шаблон проектирования, представляющий действие. Объект команды также заключает в себе само действие и его параметры. По заданию необходимо реализовать математические операции, а также возможность их исполнения и отмены. Все операции реализовываются с помощью создания объектов-команд.

Шаблон **Decorator** позволяет расширять функциональность объектов за счет применения «декораторов», добавляющих новые свойства. Так, в задании было необходимо изменить комплектацию уже созданного объекта-автомобиля. В качестве изменяемого свойства является описание (для простоты – строка, которая изменяется от декоратора к декоратору).

**Выводы**

В процессе выполнения практической работы были изучены 10 шаблонов GoF. Некоторые из них в каком-то виде напоминают общий шаблоны GRASP, однако они имеют более чёткое описание и назначение. Особенно часто используемыми шаблонами видятся шаблоны Абстрактная фабрика и Фабричный метод, т.к. почти в каждой системе существует функционал создания новых объектов. Поэтому при реализации проекта домашнего задания на эти шаблоны будет обращено особое внимание.

При выполнении заданий (в части Фабричный метод) следующий момент пособия вызвал вопрос:



1. *Выдержка из пособия*

Не совсем ясно, почему в методе для создания подкласса используется не передаваемый параметр, а поле – опечатка?