электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.021

ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРИЛОЖЕНИЙ БАЗ ДАННЫХ НА ОСНОВЕ ШАБЛОНОВ ПРОЕКТИРОВАНИЯ

Горбунова И.С., *студент*
кафедра «Система обработки информации и управление»
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

Научный руководитель: Виноградова М.В.,к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
chernen@bmstu.ru

База данных (БД) – это неотъемлемая часть современного мира, ее применение можно встретить повсеместно. Для обработки разного рода информации требуется создание большого количества хранилищ различного назначения. В настоящее время требуется оперировать огромным объемом данных, которые требуется сохранять. Указанные данные могут иметь сложную структуру и при этом общий вид данных должен удовлетворять различным требованиям пользователей. Целью любой информационной системы является обработка данных об объектах реального мира.

Для реализации работы с БД требуется создать интерфейс пользователя, который позволит эффективно выполнять все требуемые операции. Несмотря на отличия баз данных по предметной области использования, по хранящимся объектом и их свойствам, по методам и способам работы с ними, есть некоторые базовые функции для выполнения простейших операций с объектом, а именно: добавление данных, изменение, удаление, поиск. Если для каждого объекта полностью реализовать все свои собственные функции, то получим повторное использование кода, чего желательно избегать.

В настоящее время существует много технологий разработки баз данных, в том числе эти [5]. Но для упрощения и создания более конкретного каркаса предлагается создать такой шаблон для реализации программы, который бы являлся каркасом для разработки будущей системы [4]. Такой каркас должен описывать общие операции работы с объектами, как в графическом интерфейсе, так и на уровне БД. При этом функции оперируют не конкретными объектами, а абстрактными, что позволит в дальнейшем добавлять нужные классы без особых усилий. Таким образом, созданный каркас позволит уменьшить трудозатраты по разработке итоговой программы. Для реализации каркаса предлагается использование порождающих и поведенческих паттернов проектирования.

«Паттерн является структурированным описанием задачи, с которой часто сталкиваются в процессе проектирования, ее наиболее подходящее решение и, как результат, рекомендации по применению этого решения в различных ситуациях. Удачно определенный паттерн позволяет пользоваться этим решением вновь. Также паттерн можно определить как описание взаимодействия объектов и классов, адаптированных для решения общей задачи проектирования в конкретном контексте» [1].

Использование паттернов предоставляет следующие преимущества:

- формирует простую и наглядную модель структуры или поведения системы;
- позволяет выполнить анализ архитектуры системы с использованием конкретного языка или платформы;
- повышает устойчивость системы к изменению требований и упрощает последующую доработку системы;
- является унифицированным средством и облегчает взаимодействие разработчиков.

Паттерны проектирования отличаются своим назначением и уровнем применения в системе, поэтому они должны быть сгруппированы определенным образом. Существует несколько способов классификации паттернов. В данной статье рассмотрим критерии, описанные в [1].

Первый критерий классификации – цель, т.е. основная функция паттерна для реализации рассматриваемой системы. В связи с этим выделяют:

- порождающие паттерны, которые относятся к процессу создания объектов;
- структурные паттерны, которые характеризуют структуру взаимодействия объектов и классов;
- паттерны поведения, которые описывают взаимодействие классов или объектов между собой.

Второй критерий – уровень абстракции паттерна, т.е. определение цели применения паттерна: к объектам или классам. Паттерны уровня классов или объектов описывают отношения между классами и их подклассами или объектами соответственно. При этом отношения между классами являются статичными, а между объектами – динамичными.

Для реализации любой задачи можно совмещать использование разных паттернов. Однако в данной статье остановимся лишь на порождающих паттернах и на паттернах поведения, с помощью которых был создан каркас программы.

Задача поставлена таким образом, что требуется реализовать графический интерфейс для работы с объектами, при этом каждой таблице из БД соответствует свой

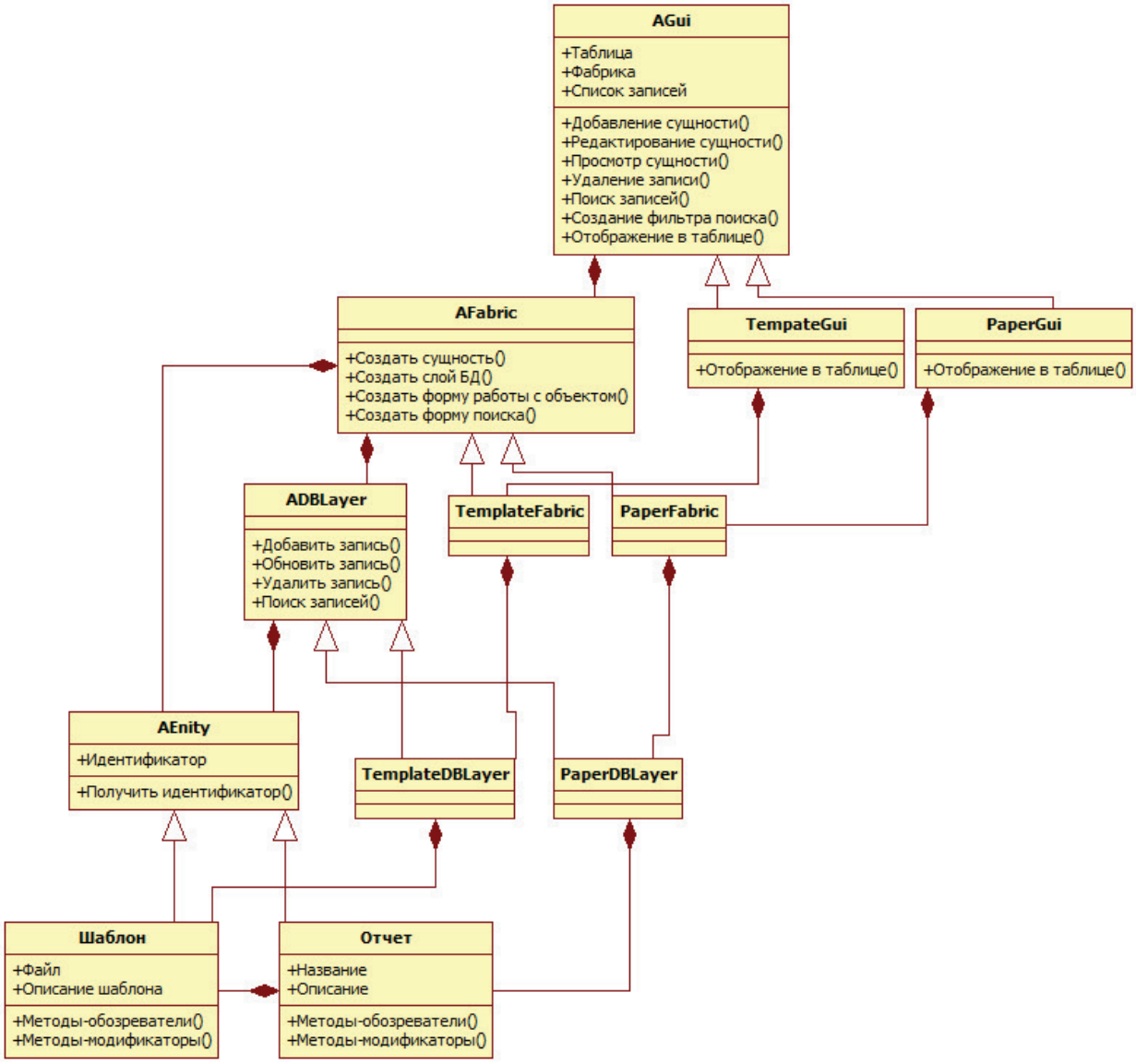
класс, а каждой записи – объект в программе. Таким образом, нужно создать каркас программы, который позволит реализовать следующие задачи:

- единообразное отображение объектов;
- однообразная работа с объектами на уровне БД, в том числе: добавление, изменение, просмотр, удаление, поиск;
- схожая работа с объектами на уровне графического интерфейса.

Для того чтобы избежать повторного использования кода, а также для уменьшения трудозатрат при создании классов для работы с разными объектами были использованы порождающие паттерны. Каркас создавался для конкретной программной системы, которая реализует интерфейс пользователя для работы с базой данных отчетов. БД содержит несколько таблиц, в которых хранятся сведения об отчетах, их форматах и шаблонах. Графический интерфейс должен обеспечивать основные функции по работе с записями таблиц.

При реализации поставленной задачи были использованы несколько видов паттернов: абстрактная фабрика и фабричный метод из группы порождающих паттернов и паттерны стратегия и шаблонный метод из группы паттернов поведения. Рассмотрим каждый паттерн и его использование в отдельности.

Полученная система классов, реализованная с использованием паттернов, представлена на рисунке.



Паттерн Абстрактная фабрика предоставляет возможность создания семейства объектов, которые связаны между собой или которые зависят друг от друга. При этом конкретные классы этих объектов не указываются, т.е. система объектов рассматривается в общем.

Паттерн абстрактная фабрика используется, когда [1]:

– «система не должна зависеть от того, как создаются, компонуются и представляются входящие в нее объекты;
– взаимосвязанные объекты должны использоваться вместе;
– требуется реализовать библиотеку объектов, не раскрывая реализацию интерфейсов, т.е. появляется возможность изолирования конкретных классов».

В качестве первого примера реализации паттерна «Абстрактная фабрика» можно привести класс «AEntity», который является абстракцией объекта в БД. Этот класс играет роль абстрактного продукта в реализации паттерна. Класс содержит атрибуты: значение

идентификатора объекта и флаг необходимости сохранения объекта в БД, а также их методы-обозреватели, которые возвращают соответствующее значение. Данный класс является родительским для всех классов, соответствующих объектам БД. Каждый объект БД должен обладать идентификатором, который определен в БД. В производных классах объектов БД – в конкретном рассматриваемом данном случае это классы «Шаблон» и «Отчет» – добавляются атрибуты, характеризующие эти объекты, а также методы-модификаторы и методы-обозреватели для работы с ними. Методы-модификаторы изменяют значение атрибута объекта и имеют только один аргумент – новое значение атрибута. Методы-обозреватели возвращают текущее значение атрибута объекта и не имеют аргументов. Часто названия этих методов начинаются с Get и Set соответственно. Классы «Шаблон» и «Отчет» играют роли конкретных объектов в реализации паттерна и относятся к разным семействам продуктов.

Следующий представитель данного паттерна – это класс «ADBLayer», который определяет набор базовых виртуальных функций для выполнения простых операций над объектом на уровне БД. Он также играет роль абстрактного продукта в реализации паттерна. Класс содержит методы добавления, удаления, изменения записей, а также выбора объектов по фильтру. При этом все функции класса являются абстрактными, т.е. их требуется определять в классах-наследниках: «TemplateDBLayer» и «PaperDBLayer», поскольку действия по отображению объектов на записи таблиц БД весьма конкретны. Классы «TemplateDBLayer» и «PaperDBLayer» играют роль конкретных объектов в реализации паттерна и относятся к разным семействам продуктов. В родительском классе «ADBLayer» в качестве входного параметра имеющихся функций выступает указатель на абстрактный объект класса «AEntity». В наследуемых классах при реализации функций используется уже конкретный объект («Шаблон» или «Отчет»), что достигается с помощью приведения типов.

Последний пример реализации данного вида паттерна – это класс «AFabric», являющийся абстрактной фабрикой. Класс содержит набор виртуальных функций для создания семейства абстрактных продуктов. Все функции являются абстрактными, и их требуется переопределять. Наследуемыми классами являются «TemplateFabric» и «PaperFabric», которые играют роль конкретных фабрик. Каждый из них обеспечивает создание конкретного семейства продуктов. В данном случае это семейства шаблонов и отчетов. Функция абстрактного класса «Создать сущность» в качестве выходного параметра возвращает указатель на абстрактный объект «AEntity», а в классах-наследниках возвращается объект конкретного класса. Аналогично реализована функция «Создать слой БД». Функция «Создать форму работы с объектом» получает в качестве

параметра указатель на абстрактный объект класса «AEntity». Функция «Создать форму поиска» не использует в качестве параметров абстрактные классы, однако ее надо полностью переопределять из-за того, что требуется вызывать разные формы поиска, которые значительно отличаются от объекта к объекту из-за того, что все объекты отличаются по количеству и содержанию полей данных.

Таким образом, были рассмотрены три класса, которые использовались в реализации паттерна «Абстрактная фабрика». Далее рассмотрим применение другого типа порождающих паттернов – фабричный метод.

Фабричные методы предоставляют проектировщику возможность не встраивать в код программы классы, которые зависят от приложения. В данном случае имеется в виду, что код использует лишь интерфейс абстрактного класса-родителя, при этом сами функции могут быть определены совершенным разными способами в классах-наследниках, которые являются конкретными продуктами.

В данном паттерне выполняется правило: «Создавай объекты в отдельной операции, чтобы подклассы могли подменить способ их создания» [1]. Данное правило дает возможность изменять классы объектов так, как необходимо.

Фабричные методы могут быть как без аргументов, так и с передаваемым параметром. При этом входной параметр определяет вид создаваемого объекта, который разделяет интерфейс главного класса.

Паттерн фабричный метод используется, когда:

- заранее неизвестно, какие объекты потребуются для работы программы;
- класс спланирован таким образом, что создаваемые объекты описываются подклассами:
- главный класс отдельно не используется, а он передает свои обязанности одному из наследуемых подклассов.

В представленной системе классов фабричные методы используются для создания объектов, используемых в шаблоне графического интерфейса. Класс «AGui», который содержит шаблоны алгоритмов для работы с объектами БД, обращается к фабрике для создания конкретного продукта. Фабричные методы абстрактной фабрики создают и возвращают конкретные продукты в зависимости от того, какая именно фабрика используется. При этом вызывающие объекты не знают ни конкретный класс фабрики, ни конкретный класс получаемых продуктов.

Кроме абстрактной фабрики фабричные методы определены в классе работы с БД «ADBLayer», в котором создают и возвращают фильтр для поиска по конкретной таблице БД, и в классе «AGui», в котором используются для создания объекта на основе

выбранной по виджету таблицы записи. В обоих случаях базовые классы определяют интерфейс фабричных методов по созданию требуемых объектов, а производные классы реализуют их по своим потребностям.

Паттерн Стратегия предоставляет возможность создания семейства алгоритмов, которые являются взаимозаменяемыми или могут предоставлять разные вариации одного и того же метода. При этом существует возможность независимого изменения самого семейства алгоритмов, т.е. стратегии.

Паттерн стратегия используется, когда:

- существуют классы, которые похожи друг на друга, но различаются только реализацией имеющихся методов. Стратегия дает возможность спроектировать класс так, что будет использоваться одно из возможных поведений;
- требуется иметь несколько разных вариантов реализации алгоритма, что применяется, когда варианты алгоритмов реализованы в виде иерархии классов или когда имеется разветвленная система поведения;
- требуется не раскрывать особенности структуры данных алгоритма.

На диаграмме классов, изображенной на рисунке, ярким представителем данного вида паттернов является класс «AGui». Это абстрактный класс шаблона графического интерфейса для работы с объектами. Класс содержит набор функций, как абстрактных, так и реализованных, для работы с таблицей объектов. Этот класс играет роль контекста в реализации паттерна.

Атрибутами данного класса являются виджет таблицы, указатель на объект фабрики «AFabric» и список объектов, извлеченных из БД – List<AEntity*>. Соответственно для каждого атрибута реализованы методы-обозреватели, а их значения устанавливаются в конструкторе класса. Объекты фабрики играют роли конкретных стратегий и через фабричные методы позволяют создавать конкретные объекты работы с БД, которые использует контекст «AGui».

Кроме упомянутых функций в «AGui» реализован также полный набор операций для работы с объектами и записями объектов на форме: данные отображаются в таблице списком или каждый объект можно посмотреть в отдельной форме. Так, были реализованы функции «Добавление сущности», «Редактирование сущности», «Просмотр сущности», «Удаление записи», «Поиск записей» и «Создание фильтра поиска». Данные операции реализованы в классе «AGui» таким образом, что они оперируют абстрактными объектами ADBLayer, AFabric и AEntity.

Класс «ADBLayer» и наследуемые от него «TemplateDBLayer» и «PaperDBLayer» играют роли абстрактной и конкретных стратегий. Класс «ADBLayer» объявляет общий

для всех поддерживаемых алгоритмов интерфейс. Однако, в нашем случае, данный класс определяет лишь набор функций, а реализация каждого алгоритма находится уже в конкретном классе. Это происходит из-за того, что алгоритмы значительно отличаются по используемым значениям и объектам. Класс «AGui» пользуется интерфейсом для вызова конкретного алгоритма, определенного в классе «TemplateDBLayer» или «PaperDBLayer».

Паттерн Шаблонный метод позволяет сохранять основную структуру алгоритма при переопределении некоторых функций в классах-наследниках.

Паттерн шаблонный метод используется, когда:

- требуется однократно проходить симметричные части алгоритма, при этом отличающиеся части поведения реализованы в подклассах;
- когда все подклассы обладают общим поведением, лишь незначительно различающимся друг от друга, т.е. необходимо выделить общее в коде и это реализовать в классе-родителе, а различия выносятся в отдельные операции, которые и будут шаблонными методами.

Шаблонные методы вызывают операции следующих видов [1]:

- конкретные операции (либо из класса ConcreteClass, либо из классов клиента);
- конкретные операции из класса AbstractClass (т.е. операции, полезные всем подклассам);
- примитивные операции (т.е. абстрактные операции);
- фабричные методы (паттерн фабричный метод);
- операции-зацепки (hook operations), реализующие поведение по умолчанию, которое может быть расширено в подклассах.

Представителем данного вида паттернов также является класс «AGui». В классе есть абстрактная функция – «Отображение в таблице», которую следует переопределять в классах-наследниках. Это связано с тем, что у каждого объекта разный набор полей, поэтому надо отображать разное количество столбцов в таблице с разными записями в ее шапке. Вызов функции происходит в шаблонных методах по работе с записями, определенных в базовом классе.

Рассмотрим реализацию данного паттерна на примере класса «AGui» и классов-наследников «TemplateGui» и «PaperGui». AGui – абстрактный класс, который определяет операции, замещаемые в конкретных подклассах для реализации шагов алгоритма. Также он реализует шаблонный метод, определяющий скелет алгоритма. Шаблонный метод вызывает примитивные операции, а также операции, определенные в классе «AGui». В качестве примитивных операций выступают функции работы с сущностями: добавление, изменение, просмотр, поиск, которые единообразны для всех объектов.

Конкретные классы «TemplateGui» и «PaperGui» реализуют операции, выполняющие шаги алгоритма способом, который зависит от подкласса. В нашем примере существует только одна такая операция – «Отображение в таблице», которая полностью зависит от текущего объекта и реализована в конкретных классах.

Исходя из диаграммы классов, изображенной на рисунке, можно выделить следующие семейства абстрактных продуктов: это классы AEntity, ADBlayer, AFabric и AGui. Данные объекты входят в иерархии связанных классов. Так, объект AEntity находится на самом нижнем уровне, но для полноценной работы программы требуется его использование во всех остальных абстрактных классах. В качестве конкретных продуктов выступают классы-наследники от каждого абстрактного продукта, а именно объекты Шаблон и Отчет, TemplateDBlayer и PaperDBlayer, TemplateFabric и PaperFabric, TemplateGui и PaperGui соответственно.

Таким образом, для реализации работы с объектами были использованы паттерны для обобщенности работы с классами, а также для четкого разграничения работы различных уровней. Все функции были разделены и выделены в отдельные базовые абстрактные классы, чтобы при необходимости не надо было менять код основной программы при добавлении новых данных, а также при изменении существующих.

Список литературы

1. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. 368 с.
2. Дубина О. Обзор паттернов проектирования // ЦИТФорум. 2005. URL: http://citforum.ru/SE/project/pattern/index.shtml#toc (дата обращения 15.06.2012).
3. Федоров О. Шаблоны проектирования: практические примеры // Программист о программировании. 2010. URL: http://glan-saratov.ru/category/шаблоны-проектирования/ (дата обращения 20.02.2013).
4. Виноградова М.В. Мазнев В. Г.. Автоматизация разработки интерфейсов взаимодействия для обмена данными с устройствами // Проблемы построения и эксплуатации систем обработки информации и управления: Сб. статей (М.) под ред. Черненького В.М. – 2007. Выпуск 5.
5. Виноградова М.В., Гжельская М.О. Технология проектирования баз данных на примере пропускной системы общежития // Проблемы построения и эксплуатации систем обработки информации и управления: Сб. статей (М.) под ред. Черненького В.М.. – 2011. Выпуск 8.