МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики
Отделение интеллектуальных кибернетических систем

Лабораторная работа №3

по курсу «Провектирование информационных систем»

Вариант 1.

Подписи:

Исполнитель

студент гр. ИС-Б17 _____ В. Ю. Петренко

Принял

д.т.н., профессор _____ Н. Л. Сальников

Обнинск, 2020

Тема: Описание проектируемой системы с помощью UML диаграммы классов (class diagram).

Вариант 1 - Система скидок (сезонные, постоянным клиентам) в магазинах (паттерн «Стратегия»).

Диаграмма классов (class diagram) – диаграмма, на которой представлена совокупность элементов модели, таких как классы с атрибутами и операциями, а также связывающие их отношения. На диаграмме классов есть статическая модель, то есть на ней не отображаются временные процессы.

Класс (class) – абстрактное описание множества однородных объектов, имеющих одинаковые атрибуты, операции и отношения с объектами других классов. Класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции. В этих секциях могут указываться имя класса, атрибуты и операции класса.

Класс может иметь или не иметь экземпляров или объектов. В зависимости от этого в языке UML различают конкретные и абстрактные классы – без экземпляров (и интерейсы). Для обозначения имени абстрактного класса используется наклонный шрифт (курсив). В языке UML принято общее соглашение о том, что любой текст, относящийся к абстрактному элементу, записывается курсивом. Для этой цели используется специальный символ разделитель – двойное двоеточие – (::). Синтаксис строки имени класса в этом случае будет следующий: <Имя пакета>::<Имя класса>.

Атрибут (attribute) – отдельное свойство или признак, который является общим для всех объектов данного класса. В языке UML принята определенная стандартизация записи атрибутов класса, которая подчиняется некоторым синтаксическим правилам. Каждому атрибуту класса соответствует отдельная строка текста, которая состоит из квантора видимости атрибута, имени атрибута, его кратности, типа значений атрибута и, возможно, его исходного значения. Общий формат записи отдельного атрибута класса следующий:<квантор видимости> <имя атрибута> [кратность] :<тип атрибута> = <исходное значение> {строка-свойство}. Видимость: "+" – public. "#" – protected. "-" – private. "~" - package. Квантор видимости может быть опущен. Имя атрибута –

единственный обязательный элемент синтаксического обозначения атрибута. Тип атрибута – тип в языке программирования.

Производный атрибут (derived element) – атрибут, значение которого для отдельных объектов может быть вычислено посредством значений других атрибутов этого же объекта.

Каждой операции класса соответствует отдельная строка, которая состоит из квантора видимости операции, имени операции, выражения типа возвращаемого операцией значения и, возможно, строка-свойство данной операции. Общий формат записи отдельной операции класса следующий: <квантор видимости> <имя операции>(список параметров):<выражение типа возвращаемого значения>.

Имя операции – единственный обязательный элемент синтаксического обозначения операции, должно начинаться со строчной (малой) буквы, и, как правило, записываться без пробелов.

Список параметров является перечнем разделенных запятой формальных параметров, каждый из которых, в свою очередь, может быть представлен в следующем виде: <направление параметра> <имя параметра>: <выражение типа> = <значение параметра по умолчанию>.

Направление параметра – есть одно из ключевых слов in, out или inout со значением in по умолчанию, в случае если вид параметра не указывается. Операция с областью действия на весь класс показывается подчеркиванием имени и строки выражения типа (static).

Управляющий класс (control class) – класс, отвечающий за координацию действий других классов. На каждой диаграмме классов должен быть хотя бы один управляющий класс. Как правило, данный класс является активным и инициирует рассылку множества сообщений другим классам модели. Кроме специального обозначения управляющий класс может быть изображён в форме прямоугольника класса со стереотипом <<control>>.

Класс-сущность (entity class) – пассивный класс, информация о котором должна храниться постоянно и не уничтожаться с выключением системы. Как правило, этот класс соответствует отдельной таблице базы данных. Класс-сущность может быть изображён также стандартным образом в форме прямоугольника класса со стереотипом <<entity>>.

Граничный класс (boundary class) – класс, который располагается на границе системы с внешней средой и непосредственно взаимодействует с актерами, но является составной частью системы. Граничный класс может быть изображён также стандартным образом в форме прямоугольника класса со стереотипом <<boundary>>.

Интерфейс в контексте языка UML является специальным случаем класса, у которого имеются операции, но отсутствуют атрибуты. Для обозначения интерфейса

используется специальный графический символ окружность или стандартный способ – прямоугольник класса со стереотипом <<interface>>.

Интерфейсы на диаграмме служат для спецификации таких элементов модели, которые видимы извне, но их внутренняя структура остается скрытой от клиентов.

Базовые отношения, изображаемые на диаграммах классов:

- отношение ассоциации (association relationship);
- отношение обобщения (generalization relationship);
- отношение агрегации (aggregation relationship);
- отношение композиции (composition relationship).

Ассоциация (association) – соответствует наличию произвольного отношения или взаимосвязи между классами. Обозначается сплошной линией со стрелкой или без нее и с дополнительными символами – имя ассоциации, символ навигации, а также имена и кратность классов-ролей ассоциации.

Имя ассоциации – необязательный элемент её обозначения.

Направленная бинарная ассоциация изображается сплошной линией с простой стрелкой. Направление этой стрелки указывает на то, какой класс является первым, а какой – вторым.

Частный случай отношения ассоциации – исключающая ассоциация (Xorassociation). Из нескольких потенциально возможных вариантов данной ассоциации в каждый момент времени может использоваться только один. Исключающая ассоциация изображается пунктирной линией, соединяющей две и более ассоциации, рядом с которой записывается в фигурных скобках: {xor}.

n-арная ассоциация (n-ary association) – ассоциация между тремя и большим числом классов. Такая ассоциация связывает отношением более чем три класса, при этом класс может участвовать в ассоциации более чем один раз. Графически n-арная ассоциация обозначается ромбом, от которого ведут линии к символам классов данной ассоциации.

Конец ассоциации (association end) – концевая точка ассоциации, которая связывает ассоциацию с классификатором. Конец ассоциации – часть ассоциации, но не класса. Одним из примеров конца ассоциации является имя роли отдельного класса, входящего в ассоциацию.

Кратность ассоциации означает потенциальное число отдельных экземпляров класса, которые могут иметь место, когда остальные экземпляры или объекты классов фиксированы.

Ассоциация является наиболее общей формой отношения в языке UML. Все другие типы рассматриваемых отношений можно считать частным случаем данного отношения.

Отношение обобщения является обычным отношением классификации между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком).

Согласно одному из главных принципов методологии ООАП – наследованию, класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет собственные свойства и поведение, которые могут отсутствовать у класса-предка.

На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов. Стрелка указывает на более общий класс (класс-предок или суперкласс), а её начало – на более специальный класс (класс-потомок или подкласс).

В дополнение к простой стрелке обобщения может быть присоединена строка текста, указывающая на специальные свойства этого отношения в форме ограничения.

В качестве ограничений могут быть использованы следующие ключевые слова UML:

- {complete} – означает, что в данном отношении обобщения специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может;

- {incomplete} – означает случай, противоположный первому;
- {disjoint} – означает, что классы-потомки не могут содержать объектов, одновременно являющихся экземплярами двух или более классов;
- {overlapping} – случай, противоположный предыдущему. А именно,
- предполагается, что отдельные экземпляры классов-потомков могут принадлежать одновременно нескольким классам.

Агрегация (aggregation) – служит для представления отношения типа "частьцелое" между агрегатом (целое) и его составной частью.

Отношение агрегации показывает, из каких элементов состоит система, и как они связаны между собой. Данное отношение по своей сути описывает декомпозицию или разбиение сложной системы на более простые составные части, которые также могут быть подвергнуты декомпозиции.

Отношение агрегации изображается сплошной линией, один из концов которой представляет собой не закрашенный внутри ромб. Этот ромб указывает на тот класс, который представляет собой "целое" или класс-контейнер.

Композиция (composition) – разновидность отношения агрегации, при которой составные части целого имеют такое же время жизни, что и само целое. Эти части уничтожаются вместе с уничтожением целого. Отношение композиции – частный случай отношения агрегации. Части не могут выступать в отрыве от целого.

Отношение композиции изобра жается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот класс, который представляет собой класс-композит. Остальные классы являются его «частями».

Паттерн «Стратегия».

Паттерн «Стратегия» переносит в отдельную иерархию классов все детали, связанные с реализацией алгоритмов. Абстрактный базовый класс иерархии объявляет интерфейс, общий для всех алгоритмов. Подклассы реализуют его в соответствии с тем или иным алгоритмом. Класс содержит указатель на объект абстрактного типа и предназначен для переадресации пользовательских запросов конкретному алгоритму. Для замены одного алгоритма другим достаточно перенастроить этот указатель на объект нужного типа. Достоинства паттерна «Стратегия»:

систему проще поддерживать и модифицировать, так как семейство алгоритмов перенесено в отдельную иерархию классов;

паттерн «Стратегия» предоставляет возможность замены одного алгоритма другим в процессе выполнения программы;

паттерн «Стратегия» позволяет скрыть детали реализации алгоритмов от клиента.

Недостатки паттерна «Стратегия»:

для правильной настройки системы пользователь должен знать об особенностях всех алгоритмов;

число классов в системе, построенной с применением паттерна «Стратегия», возрастает.

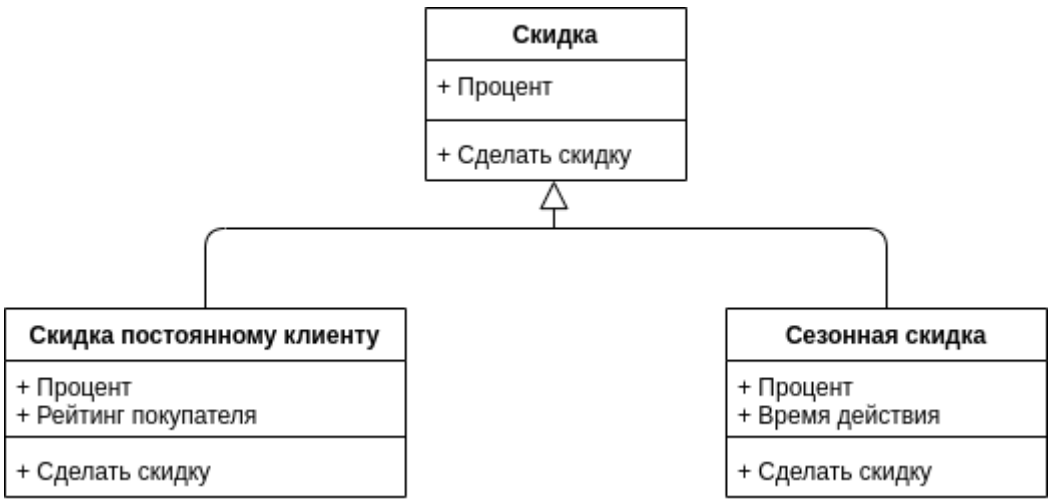

Рисунок 1 — Диаграмма классов системы скидок