Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Пермская государственная сельскохозяйственная академия
имени академика Д.Н. Прянишникова»

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

направление 230700 «Прикладная информатика»

ЛАБОРАТОРНОЕ ЗАНЯТИЕ № 6

Тема: **Модель предметной области**

Учебные вопросы:

1. Принципы создания модели предметной области.
2. Модель предметной области: добавление ассоциаций и атрибутов.

Вопрос 1. Принципы создания модели предметной области

Модель предметной области широко используется в качестве основы для разработки программных объектов и обеспечивает важную входную информацию для создания нескольких последующих артефактов.

Модель предметной области отображает основные (с точки зрения моделирующего) классы понятий (концептуальные классы) предметной области. Она является наиболее важным артефактом, создаваемым на этапе объектно-ориентированного анализа[1]. Основной задачей объектно-ориентированного анализа является идентификация большого количества разнообразных объектов или понятий, а также точная оценка усилий в терминах отдачи на стадиях проектирования и реализации.

Идентификация классов понятий или концептуальных классов – составная часть исследования предметной области. Модели предметной области на языке UML строятся в форме диаграмм классов.

Для создания модели предметной области выполните следующие действия:

1. Составьте список кандидатов на роль концептуальных классов на основе списка категорий и метода анализа текстового описания для текущей итерации разработки.
2. Отобразите их в модели предметной области.
3. Добавьте необходимые ассоциации, отражающие связи, для которых требуется выделение памяти.
4. Добавьте атрибуты, необходимые для выполнения информационных требований.

Имена и модели: стратегия построения карт

При построении моделей предметной области применяется та же стратегия, что и при создании карт.

Модель предметной области следует создавать согласно принципам картографии:

- использовать применяемые на данной территории названия;
- исключать несущественные детали;
- не добавлять объекты, которые отсутствуют на данной территории.

Модель предметной области – это своеобразная разновидность карты понятий некоторой предметной области. Отсюда следуют аналитическая роль модели предметной области, а также справедливость следующих замечаний:

• Картографы используют названия, применяемые на данной территории. Они не изменяют названия городов на карте. С точки зрения модели предметной области это означает необходимость **использования словаря предметной области при именовании концептуальных классов и атрибутов**. Например, при разработке модели библиотеки в качестве понятия, означающего потребителя, следует выбрать Borrower (Читатель), т.е. использовать терминологию библиотекарей.

• Картограф не наносит на карту объекты, не имеющие отношения к основному ее назначению, например не отображает топографию или состав населения. Аналогично, в модели предметной области **не должны содержаться понятия из предметной области, не имеющие отношения к требованиям**. Например, из нашей модели предметной области можно исключить понятия Pen (Ручка) и PaperBag (Папка), поскольку они не имеют отношения к требованиям.

• Картограф не отображает на карте отсутствующие объекты, например, горы на карте равнинной местности. Точно так же, в модели предметной области **не должны содержаться понятия, не имеющие отношения к рассматриваемой проблеме.**

Этот принцип называют также стратегией использования словаря предметной области.

---

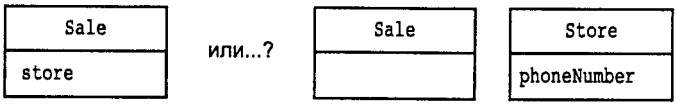
[1] Прецеденты — это важные артефакты этапа анализа требований, но на самом деле они не являются объектно-ориентированными. Они концентрируют внимание разработчиков на процессах, происходящих в предметной области.

Типичная ошибка при выделении концептуальных классов

Возможно, наиболее типичной ошибкой при создании модели предметной области является отнесение некоторого объекта к атрибутам, в то время как он должен относиться к концептуальным классам. Чтобы избежать этой ошибки, следует придерживаться простого правила.

Если некоторый объект X в реальном мире не является числом или текстом, значит, это скорее концептуальный класс, чем атрибут.

Например, является ли store (магазин) атрибутом объекта Sale (Продажа) или отдельным концептуальным классом Store?



В реальном мире магазин не является числом или текстом, он представляет реальную сущность, организацию, занимающую некоторое место. Следовательно, Store нужно рассматривать в качестве концептуального класса.

Необходимость спецификаций или описание концептуальных классов

Описанная проблема иллюстрирует необходимость использования понятий, представляющих собой спецификации или описание других объектов. Для решения этой проблемы необходимо ввести концептуальный класс ProductSpecification (Спецификация товара) (или itemSpecification, ProductDescription и т.д.), содержащий информацию о товаре. Понятие ProductSpecification не совпадает с Понятием Item, а представляет собой лишь описание этого товара. Заметим, что даже после продажи всех единиц товара и удаления из памяти всех программных элементов Item понятие ProductSpecification остается в памяти компьютера.

Объекты описаний или спецификации тесно связаны с описываемыми понятиями. В модели предметной области связь между ними принято обозначать следующим образом: XSpecification описывает X (рис.1.1).

Понятия-спецификации обычно используются при описании предметной области, связанной с продажами или товарами. Они также часто используются при описании производства, для обозначения отличий одних производимых товаров от других. Эти понятия идентифицируются одними из первых, поскольку являются очень типичными.

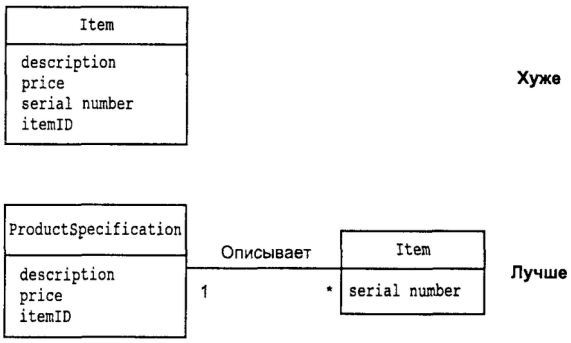


Рисунок 1.1 – Спецификации или описания других понятий

Символ * означает "множественную" связь и указывает на то, что одно понятие Product-Specification может описывать много (*) понятий Item.

Когда требуются понятия-спецификации

Рассматривая вопрос о необходимости введения понятий-спецификаций, нужно принимать во внимание следующие соображения.

Концептуальный класс спецификации или описания (например, ProductSpecification) вводится в таких случаях:

- Существует необходимость описания элементов или служб независимо от существования

конкретных экземпляров этих объектов.

- Если удаление экземпляров описываемых им понятий (например, Item) приводит к потере важной информации в связи с некорректной ассоциацией этой информации с удаляемым экземпляром.
- Если при наличии понятия устраняется дублирование информации.

Пример: модель предметной области POS-системы ТТ

Список концептуальных классов предметной области POS-системы ТТ можно представить графически (рис. 1.2) в рамках модели предметной области.

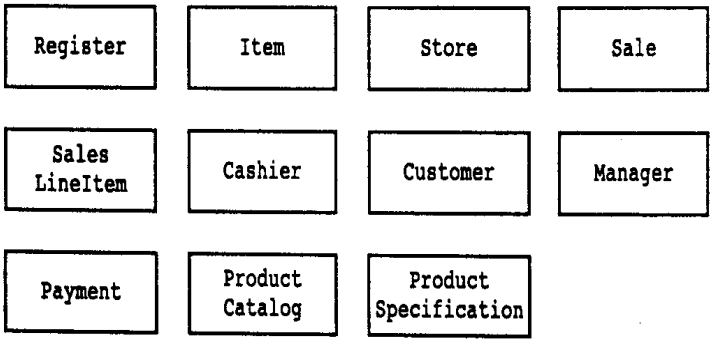


Рисунок 1.2 – Исходная модель предметной области

Концептуальные классы

Модель предметной области иллюстрирует концептуальные классы или словарь предметной области.

Неформально, **концептуальный класс** – это представление идеи или объекта. Если говорить более строго, то концептуальный класс можно рассматривать в терминах символов, содержания и расширения (рис. 1.3):

- *Символы* (symbol) – слова или образы, представляющие концептуальный класс.
- *Содержание* (intension) – определение концептуального класса.
- *Расширение* (extension) – набор примеров, к которым применим концептуальный класс.

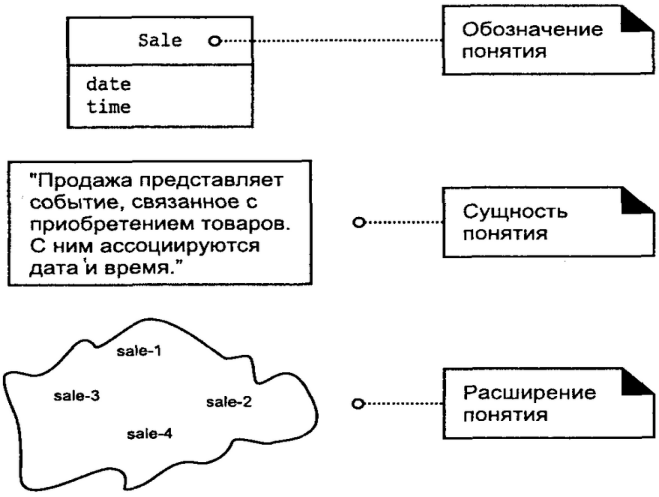


Рисунок 1.3 – Символ, содержание и расширение концептуального класса

Например, рассмотрим концептуальный класс для события осуществления покупки. Его можно обозначить символом Sale. Содержанием этого понятия является "представление события осуществления покупки в определенный день и определенное время". В качестве расширения можно рассматривать все примеры покупок (другими словами, все множество покупок).

При создании модели предметной области обычно рассматриваются символьное описание и содержание концептуального класса, поскольку именно они представляют наибольший практический интерес.

Модели предметной области и декомпозиция

Программные системы могут быть очень сложными. Поэтому декомпозиция (по принципу "разделяй и властвуй") – это общая стратегия борьбы со сложностью проблемы за счет ее разделения на мелкие составные части. При *структурном подходе к* проектированию ИС задача разбивается на процессы или функции, а при объектно-ориентированном – на основные понятия или сущности предметной области.

Основное отличие объектно-ориентированного анализа от структурного состоит в декомпозиции проблемы на понятия (объекты), а не на функции.

Следовательно, основной задачей на стадии анализа является идентификация различных понятий из предметной области и представление результатов в виде модели предметной области.

Концептуальные классы предметной области торговли

Например, в предметной области розничной торговли можно выделить концептуальные классы Store (Магазин), Register (Реестр) и Sale (Продажа). Следовательно, модель предметной области системы автоматизации розничной торговли должна включать элементы, представленные на рис. 1.4.

Рисунок 1.4 – Фрагмент модели предметной области для системы розничной торговли

Идентификация концептуальных классов

Нашей задачей является создание модели предметной области, отражающей важные понятия рассматриваемой предметной области розничной торговли. В данном случае речь идет о понятиях, связанных с прецедентом Оформление продажи.

При итеративной разработке модель предметной области строится в течение нескольких итераций фазы развития. На каждой итерации к модели добавляются концептуальные классы рассматриваемых сценариев, а не делается попытка "объять необъятное" и сразу же построить исчерпывающую модель всех возможных концептуальных классов и их взаимоотношений.

Например, если на данной итерации рассматривается упрощенный сценарий прецедента Оформление продажи, описывающий продажу товаров за наличный расчет, то в разрабатываемом фрагменте модели предметной области нужно отобразить понятия, относящиеся только к этому сценарию.

Основная задача – идентифицировать концептуальные классы, связанные с разрабатываемым сценарием.

При идентификации концептуальных классов целесообразно руководствоваться следующим принципом:

Лучше излишне детализировать модель предметной области, чем недоопределить ее.

Зачастую на начальной стадии идентификации некоторые концептуальные классы упускаются из виду, а появляются позднее, при рассмотрении атрибутов и ассоциаций, или даже на стадии проектирования. Обнаруженные новые понятия добавляются в модель предметной области.

Не следует исключать из рассмотрения концептуальные классы только на том основании, что из анализа требований не следует очевидная необходимость их запоминания (этот критерий зачастую применяется при разработке реляционных баз данных, однако не годится для создания моделей предметной области) или концептуальный класс не имеет атрибутов.

Концептуальные классы без атрибутов вполне допустимы, как допустимы и концептуальные классы, играющие "поведенческую", а не информационную роль в предметной области.

Стратегии идентификации концептуальных классов

Ниже представлены два способа выявления концептуальных классов.

1. С использованием списка категорий концептуальных классов.
2. На основе выделения существительных.

Для моделирования предметной области применяется и другой прием – шаблоны анализа. **Шаблоны** – это готовые модели предметной области, созданные специалистами. Они широко описаны в литературе.

Использование списка категорий концептуальных классов

Приступая к созданию модели предметной области, целесообразно составить список кандидатов на роль концептуальных классов. В табл. 1.1 содержится множество стандартных категорий, которые обычно имеют важное значение. В этом перечне они приводятся в произвольном порядке и не упорядочены по степени важности. Примеры взяты из предметной области системы торговли и резервирования авиабилетов.

Таблица 1.1 – Список категорий концептуальных классов

Категория концептуальных классов	Примеры
Физические или материальные объекты	Register (Реестр), Airplane (Самолет)
Спецификации, элементы проектных решений или описания объектов	productspecification (Спецификация товара), FlightDescription (Описание полета)
Места	Store (Магазин), Airport (Аэропорт)
Транзакции	Sale (Продажа), Payment (Платеж), Reservation (Резервирование)
Элементы транзакций	SalesLineItem (Элемент продажи)
Роли людей	Cashier (Кассир), Pilot (Пилот)
Контейнеры других объектов	Store (Магазин), Bin (Бункер), Airplane (Самолет)
Содержимое контейнеров	Item (Элемент), Passenger (Пассажир)
Другие компьютеры или электромеханические системы, внешние по	CreditPaymentAuthorizationSystein (Система авторизации кредитных платежей), AirTrafficControl (Система управления движением)
Абстрактные понятия	Hunger (Голод), Acrophobia (Акрофобия)
Организации	SalesDepartment (Отдел Продаж), ObjectAirline (Авиалинии)
События	Sale (Продажа), payment (Платеж), Meeting (Встреча), Flight (Полет), Crash (Крушение), Landing (Приземление)
Процессы (зачастую *не* представляются в виде	SellingAproduct (Продажа продукта), BookingASeat (Бронирование места)
Правила и политика	RefundPolicy (Правила возврата товара) CancellationPolicy (Политика аннулирования заказа)
Каталоги	productCatalog (Каталог товаров), PartsCatalog (Каталог частей)
Записи финансовой, трудовой, юридической и другой деятельности	Receipt (Чек), Ledger (Гроссбух), EmploymentContract (Трудовой контракт), MaintenanceLog (Журнал обслуживания)
Финансовые инструменты и службы	Lineofcredit (Кредитная линия), Stock (Акция)
Руководства, документы, статьи, книги	DailyPricechangeList (Бюллетень ежедневного изменения цен), RepairManual (Руководство по восстановлению)

Определение концептуальных классов с помощью выявления существительных

Еще один полезный (и очень простой) прием идентификации концептуальных классов на основе лингвистического анализа. Он состоит в выделении существительных из текстовых описаний предметной области и их выборе в качестве кандидатов в концептуальные классы или атрибуты.

Этот метод следует применять с осторожностью. Между существительными и концептуальными классами нет взаимно однозначного соответствия, а слова естественного языка могут иметь несколько значений.

Тем не менее, это информация к размышлению. Для реализации подобного подхода удобно использовать развернутые описания прецедентов, например, основной сценарий прецедента Оформление продажи.

Основной успешный сценарий (или основной процесс)

1. Покупатель подходит к **кассовому аппарату POS-системы** с выбранными **товарами**.
2. **Кассир** открывает новую **продажу**.
3. **Кассир** вводит **идентификатор товара**.
4. Система записывает **наименование товара** и выдает его **описание, цену и общую стоимость**. Цена вычисляется на основе набора правил.

Кассир повторяет действия, описанные в пп. 3–4 для каждого наименования товара.

5. Система вычисляет общую стоимость покупки с **налогом**.
6. Кассир сообщает покупателю общую стоимость и предлагает оплатить покупку.
7. Покупатель оплачивает покупку, система обрабатывает **платеж**.
8. Система регистрирует **продажу** и отправляет информацию о ней внешней **бухгалтерской системе** (для обновления бухгалтерских документов и начисления **комиссионных**) и **системе складского учета** (для обновления данных).
9. Система выдает **товарный чек**.
10. Покупатель покидает магазин с чеком и товарами (если он что-то купил).

Расширения (или альтернативные потоки)

7а. Оплата наличными

1. Кассир вводит предложенную покупателем **сумму**.
2. Система вычисляет положенную **сдачу** и открывает **кассу с наличностью**.
3. Кассир складывает полученные деньги и выдает сдачу покупателю.
4. Система регистрирует платеж наличными.

Модель предметной области – это визуализация важных понятий из словаря предметной области. Откуда брать необходимые термины? Из описания прецедентов. Эти описания – богатый источник для идентификации существительных.

Одни из этих существительных могут быть представлены в виде концептуальных классов, другие могут представлять концептуальные классы, не имеющие отношения к данной итерации (например, "бухгалтерская система" или "комиссионные"), а третьи — в виде атрибутов этих концептуальных классов. Более подробная информация об отличиях между концептуальными классами и атрибутами содержится в следующем разделе и следующей главе.

Недостатком данного подхода является выразительность естественного языка. Для описания одного и того же концептуального класса или атрибута могут использоваться различные существительные, и в то же время некоторые существительные могут иметь по несколько значений. Тем не менее этот подход рекомендуется использовать в сочетании с методом выбора понятий по списку категорий.

Кандидатуры на роль концептуальных классов для предметной области торговли

Пользуясь списком категорий и методом анализа словесного описания, мы составили список кандидатур на роль концептуальных классов для предметной области розничной торговли. Он соответствует требованиям и принятым упрощениям для основного сценария прецедента Оформление продажи.

Register	ProductSpecification	Payment
Item	SalesLineItem	ProductCatalog
Store	Cashier	Manager
Sale	Customer	

Не существует понятия "правильный" список. Это просто произвольный набор абстракций и понятий из словаря предметной области, которые, по мнению разработчика модели, являются важными. Тем не менее, если для выделения концептуальных классов использовать описанные выше стратегии, то различные специалисты по моделированию составят примерно одинаковые списки.

Пример рассуждения: включать ли понятие "товарный чек" в модель

Товарный чек – это документ, принимающий участие в продаже товара и его оплате. Он может рассматриваться как концептуальный класс из предметной области. Возникает вопрос: нужно ли его включать в модель предметной области?

Для ответа на этот вопрос необходимо учитывать следующие факторы.

- Товарный чек – это своеобразный отчет о сделанной покупке. В принципе, в модель предметной области не следует включать объекты отчета, поскольку вся содержащаяся в них информация получена из других источников. Это является одной из причин исключения понятия "чек" из модели предметной области.
- Товарный чек выполняет конкретную роль при реализации бизнес-правил: обычно он обеспечивает право на возврат товара. Этот фактор говорит в пользу включения товарного чека в модель предметной области.

Поскольку возврат товара не рассматривается на данной итерации разработки, понятие Receipt (Товарный чек) не включается в модель предметной области. Оно будет включено в модель системы на той итерации разработки, на которой будет реализовываться прецедент Возврат товара.

Вопрос 2. Модель предметной области: добавление ассоциаций и атрибутов

В процессе разработки модели предметной области необходимо идентифицировать связи (ассоциации) между концептуальными классами, удовлетворяющие информационным требованиям разрабатываемых на текущей итерации сценариев, а также выделить те из них, которые способствуют лучшему пониманию модели предметной области.

Ассоциация (association) – это связь между типами (или точнее, экземплярами типов), отражающая некоторое значимое и полезное отношение между ними (рис. 2.1).

В языке UML ассоциации описываются как "семантические взаимосвязи между двумя или несколькими классификаторами и их экземплярами".

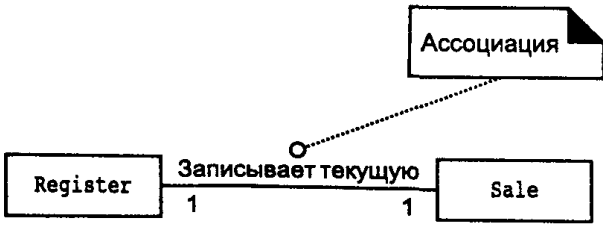


Рисунок 2.1 – Ассоциации

Поиск ассоциаций

Заслуживающие внимания ассоциации обычно содержат знания о взаимосвязи между объектами, которые должны **сохраняться** в течение некоторого периода. Этот период может измеряться в миллисекундах или годах в зависимости от конкретного контекста.

Другими словами, ***о связи между какими объектами нужно помнить***?

Например, нужно ли помнить о том, что экземпляры объекта SalesLineItem (Элемент продажи) ассоциированы с экземпляром объекта Sale (Продажа)? Очевидно да, поскольку в противном случае будет невозможно восстановить данные о продаже, распечатать товарный чек или вычислить итоговую сумму.

В модель предметной области целесообразно включать следующие ассоциации:

- ассоциации, знания о которых нужно сохранять в течение некоторого периода (важные ассоциации);

- ассоциации, производные от содержащихся в списке стандартных ассоциаций.

Это важный момент. Если в модели предметной области содержится N различных концептуальных классов, то между ними можно установить N*(N-1) ассоциацию, а это может быть достаточно большое число. Многие линии связей такой диаграммы будут просто вносить визуальный «шум» и ухудшать ее наглядность. Поэтому при добавлении ассоциаций нужно придерживаться принципа минимализма. Критерии необходимости ассоциаций будут предложены ниже.

Система обозначений для ассоциаций языка UML

Ассоциация **обозначается** проведенной между классами линией, с которой связано определенное имя. Обычно ассоциация является двунаправленной. Это означает, что от одного объекта любого типа возможен логический переход к другому объекту. Такой переход является абсолютно абстрактным. Он не определяет тип взаимосвязей между программными сущностями.

На **концах** линии, которая обозначает ассоциацию, могут содержаться выражения, определяющие количественную связь между экземплярами классов.

Дополнительная стрелка рядом с именем ассоциации указывает, в каком направлении нужно читать ее имя. Она не определяет направление видимости или перемещения.

Если такая стрелка отсутствует, то имена ассоциаций следует читать с использованием общепринятых соглашений, а именно – **слева направо и сверху вниз**. Однако в языке UML в явной форме это правило отсутствует (рис. 2.2).

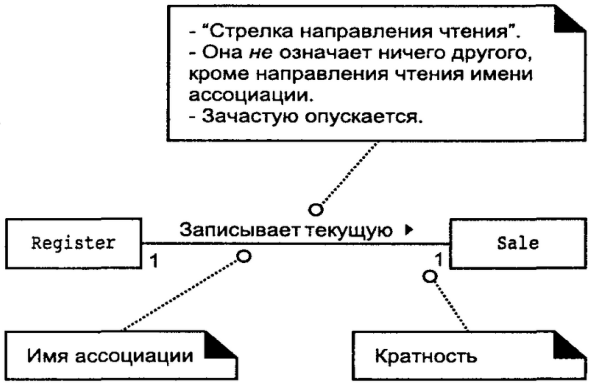


Рисунок 2.2 – Система обозначения ассоциаций в языке UML

Поиск ассоциаций: список стандартных ассоциаций

Добавим ассоциации с помощью списка, представленного в табл. 2.1. В нем указаны стандартные категории, которыми обычно не следует пренебрегать. Примеры ассоциаций взяты из предметной области розничной торговли и авиалиний.

Таблица 2.1 – Список стандартных ассоциаций

Категория	Примеры
А является физической частью В	Drawer-Register (Устройство печати торговых чеков-Реестр)
А является логической частью В	SalesLineItem-Sale (Элемент продажи-Продажа)
А физически содержится в/на В	Register-Store (Реестр-Магазин), item-shelf (Товар-Полка)
А логически содержится в В	itemDescription-Catalog (Описание товара-Каталог)
А является описанием В	itemDescription-item (Описание товара-Товар)
А является элементом транзакции или отчета В	SalesLineItem-Sale (Элемент продажи-Продажа)
А известен/ зарегистрирован/	Sale-Register(Продажа-Реестр)
А является членом В	Cashier-Store (Кассир-Магазин)

A является организационной единицей	Department-Store (Отдел-Магазин)
A использует или управляет B	Cashier-Register (Кассир-Реестр)
A взаимодействует с B	Customer-Cashier (Покупатель-Кассир)
A связан с транзакцией B	Customer-Payment (Покупатель-Платеж)
A является транзакцией, которая связана с другой транзакцией B	Payment-Sale (Платеж-Продажа)
A следует за B	salesLineitem-SalesLineitem (Наименование товара-Следующее наименование товара)
A является "собственностью" B	Register-Store (Реестр-Магазин)
A является событием, связанным с B	Sale-customer (Продажа-Покупатель), Sale-Store (Продажа-Магазин)

Ассоциации с высоким приоритетом

В приведенной таблице имеются категории ассоциаций с высоким приоритетом, которые чрезвычайно полезно включать в модель предметной области:

- A является ***физической*** (physical) или ***логической*** (logical) частью B;
- A ***физически*** или ***логически*** содержится в/на B;
- A ***записан*** (recorded) в B.

Рекомендации по назначению ассоциаций

- Основное внимание нужно уделить тем ассоциациям, знания о которых нужно сохранять в течение некоторого периода (важным ассоциациям).
- Гораздо важнее определить концептуальные классы, чем выделить ассоциации.
- Слишком большое количество ассоциаций приводит к ошибкам в модели предметной области, а не к ее упрощению. Изучение ассоциаций не должно отнимать слишком много времени и вместе с тем должно приносить максимальную пользу.
- Следует избегать отображения избыточных или не имеющих самостоятельного значения ассоциаций.

Роли

Каждый конец ассоциации называется **ролью** (role). Роль дополнительно может иметь следующие характеристики: имя, кратность, направление связи.

Кратность

Кратность (multiplicity) определяет, сколько экземпляров класса A может быть ассоциировано с одним экземпляром класса B (рис. 2.3).

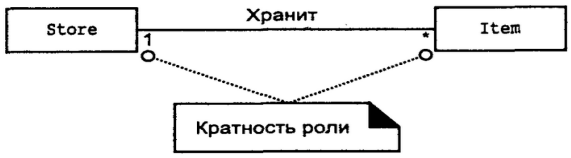


Рисунок 2.3 – Кратность ассоциации

Например, один экземпляр класса Store (Магазин) может быть ассоциирован с несколькими (ни с одним или с несколькими, что отображается символом ***)** экземплярами класса Item.

Значение кратности определяет, сколько экземпляров одного класса может быть корректно связано с экземпляром другого класса в некоторый конкретный момент, а не на всем промежутке времени.

Имена ассоциаций

Имена ассоциаций базируются на формате *ИмяТипа-ГлагольнаяФраза-ИмяТипа,* где глагольная фраза представляет собой последовательность, которая читается и является значимой в контексте модели.

Имена ассоциаций должны начинаться с прописной буквы, т.к. ассоциация обычно представляет классификатор связей между экземплярами. В языке UML имя классификатора начинается с прописной буквы. Для имен ассоциаций принято использовать два формата:

- **Paid-by (с дефисом)**
- **PaidBy (без дефиса)**

На рис. 2.4 при чтении имен ассоциаций используется направление, принятое по умолчанию, а именно – слева направо и сверху вниз. Это правило относится не только к языку UML, а является относительно стандартным.

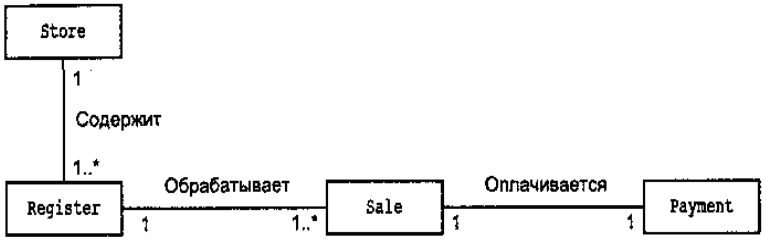


Рисунок 2.4 – Имена ассоциаций

Несколько ассоциаций между двумя типами

Между двумя типами может быть установлено несколько ассоциаций. В рассматриваемой нами POS-системе нет соответствующего примера, однако из предметной области системы управления полетами в качестве подобного примера можно привести отношение между объектами Flight (Полет) и Airport (Аэропорт) (рис. 2.5). Ассоциации Flies-to (Летит в) и Flies-from (Летит из) являются принципиально различными и должны отображаться отдельно.

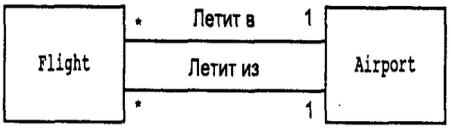


Рисунок 2.5 – Несколько ассоциаций

Ассоциации для предметной области POS-системы ТТ

Теперь приступим к добавлению ассоциаций в модель предметной области рассматриваемой POS-системы. Требуется добавить те ассоциации, которые должны сохраняться согласно определенным требованиям (например, прецедентам), или те, которые вытекают из нашего представления о предметной области. При решении новой проблемы должны быть пересмотрены и проанализированы ранее определенные **стандартные категории ассоциаций**, поскольку они представляют многие из актуальных ассоциаций, которые обычно следует учитывать.

Отношения в магазине, которые должны быть учтены

Важными являются следующие ассоциации. Они основываются на описанных выше прецедентах:

Register Records Sale (Register Фиксирует Sale)	Для того чтобы получить информацию о текущей продаже, необходимо вычислить общую сумму покупки и напечатать чек
Sale Paid-by Payment (Sale Оплачивается посредством Payment)	Чтобы узнать, была ли оплачена покупка, необходимо сравнить внесенную сумму с общей стоимостью покупки и напечатать чек
ProductCatalog Records ProductSpecification(ProductCatalog Записывает ProductSpecification)	Чтобы получить спецификацию товара ProductSpecification, необходимо знать его идентификатор itemID

Использование списка категорий ассоциаций

Проанализируем список категорий ассоциаций, основываясь на ранее идентифицированных типах и учитывая требования разрабатываемых на данной итерации прецедентов.

Категория	Примеры
A является физической частью B	Register-CashDrawer
A является логической частью B	SalesLineItern-Sale
A физически содержится в/на B	Register-Store Item-Store
A логически содержится в B	Product Speci£ication-ProductCatalog ProductCatalog-Store
A является описанием B	Product Specification-1 tern
A является элементом транзакции или отчета B	SalesLineItern-Sale
A известен/зарегистрирован/записан/ включен в B	(Совершенные покупки) Sales-Store (Текущая продажа) Sale-Register
A является членом B	Cashier-Store
A является организационной единицей B	Не применяется
A использует или управляет B	Cashier-Register Manager-Register Manager-Cashier; однако, ВОЗМОЖНО, не применяется
A взаимодействует с B	Customer-Cashier (Покупатель-Кассир)
A связан с транзакцией B	Customer-Payment Cashier-Payment
A является транзакцией, которая связана с другой транзакцией B	Payment-Sale (Платеж-Продажа)
A следует за B	SalesLineItem-SalesLineItem
A является "собственностью" B	Register-Store

Модель предметной области POS-системы TT

В показанной на рис. 2.6 модели предметной области представлен набор концептуальных классов и ассоциаций, являющихся кандидатами на включение в POS-приложение. Ассоциации предварительно были выбраны из списка ассоциаций-кандидатов.

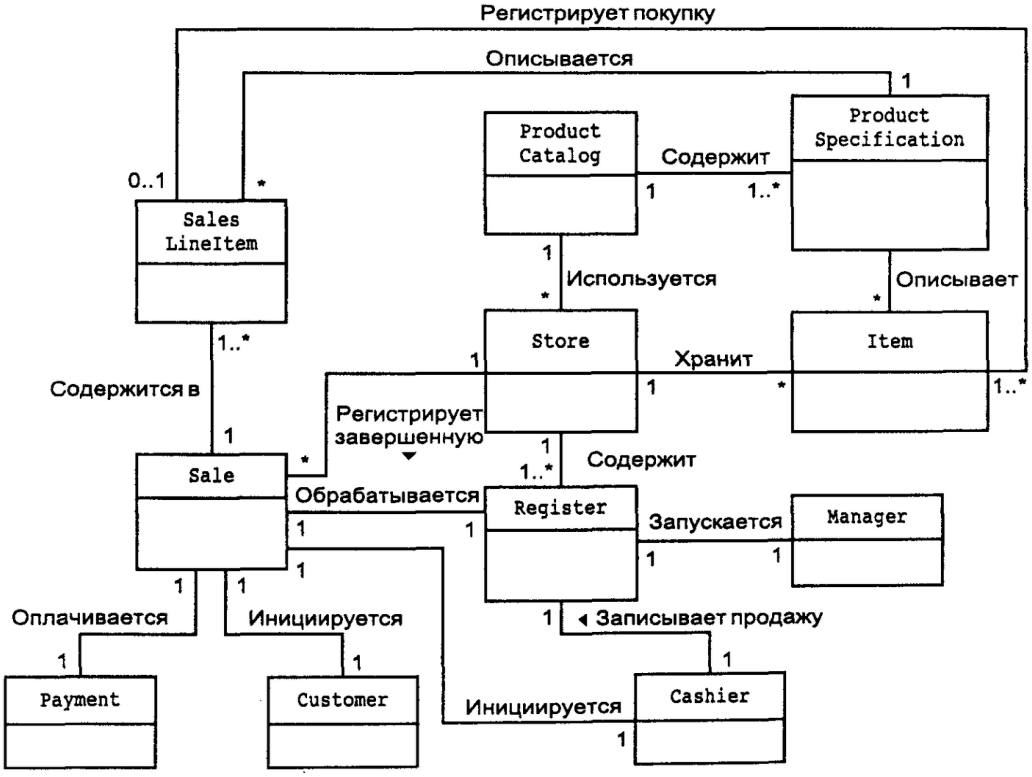


Рисунок 2.6 – Фрагмент модели предметной области

Сохранение только важных ассоциаций

Ассоциации, представленные в модели предметной области на рис. 2.6, были, по существу, механически выбраны из списка ассоциаций. Однако при включении ассоциаций в модель предметной области желательно быть более избирательным. Не стоит перегружать модель предметной области ассоциациями, которые не требуются наверняка или не отражают результатов исследования предметной области.

Из приведенных выше рекомендаций по выбору ассоциаций следует, что не каждая из представленных ассоциаций является необходимой.

Ассоциация	Описание
Sale Entered-by Cashier (Sale Вводится Cashier)	Требования не отражают необходимости хранить или записывать информацию о кассире. Кроме того, эта ассоциация напрямую следует из ассоциации Register used-by Cashier (Register Используется Cashier)
Register Used-by Cashier (Register Используется Cashier)	Требования не отражают необходимости хранить или записывать информацию о кассире
Register Started-by Manager (Register Запускается Manager)	Требования не отражают необходимости хранить или записывать информацию о менеджере, запустившем систему
Sale Initiated-by Customer (Sale Инициируется Customer)	Требования не отражают необходимости хранить или записывать информацию о текущем покупателе, инициировавшем продажу
Store Stocks Item (Store Хранит Item)	Требования не отражают необходимости хранить или поддерживать информацию об инвентаризации
SalesLineItem Records-sale-of Item (SalesLineItem Записывает-информацию-о-продаже Item)	Требования не отражают необходимости хранить или поддерживать информацию об инвентаризации

Обратите внимание, что размещение ассоциаций по степени важности зависит от требований. Внесение в них явных изменений, таких как требование наличия на товарном чеке идентификатора кассира, приведет к необходимости хранения данных о новой связи.

Основываясь на проведенном анализе, можно сказать, что эти ассоциации можно удалить.

Модель предметной области: добавление атрибутов

Необходимо идентифицировать атрибуты концептуальных классов, которые удовлетворяют информационным требованиям разрабатываемых в текущий момент сценариев.

Атрибуты

Атрибут (attribute) – это абстрактное свойство объекта.

В модель предметной области включаются те атрибуты, для которых определены соответствующие требования (например, прецеденты) или для которых необходимо хранить определенную информацию.

Например, в товарном чеке (представляющем собой отчет о некоторой продаже) обычно указываются дата и время. Эта информация необходима менеджерам компании. Следовательно, для концептуального класса Sale (Продажа) требуются атрибуты date (дата) и time (время).

Система обозначений атрибутов в языке UML

Атрибуты помещаются во второй раздел условного обозначения класса (рис. 2.7). Дополнительно может быть указан также тип атрибута.

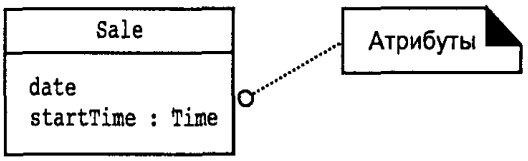


Рисунок 2.7 – Класс и его атрибуты

Корректные типы атрибутов

Некоторые сущности гораздо удобнее представлять не в виде атрибутов, а в форме ассоциаций. Изучению корректных атрибутов и посвящен данный подвопрос.

Атрибуты должны быть простыми

Типы большинства простых атрибутов зачастую рассматриваются как примитивные типы данных. Обычно тип атрибута не должен быть сложным понятием предметной области, таким как Sale (Продажа). Например, атрибут currentRegister класса Cashier (Кассир) лучше не использовать (рис. 2.8), поскольку он имеет тип Register, не являющийся простым типом атрибута (таким, как Number (Число) или String (Строка)). Если объект Cashier использует объект Register, то лучше всего выразить этот факт с помощью ассоциации, а не атрибута.

В модели предметной области **атрибуты должны быть *простыми атрибутами* (simple attributes) или *простыми типами данных* (data types).**

К стандартным типам атрибутов относятся ***Boolean, Date, Number, String(Text), Time.***

Другими стандартными типами являются следующие: **Address** (адрес), **Color** (цвет), **Geometries** (Point, Rectangle) (геометрические фигуры: точка, прямоугольник), **Phone Number** (номер телефона), **Social Security Number** (номер страхового полиса), **Universal Product Code** (UPC) (универсальный код товара), **SKU**, **ZIP** или **postal code** (почтовый индекс), перечисляемые типы.

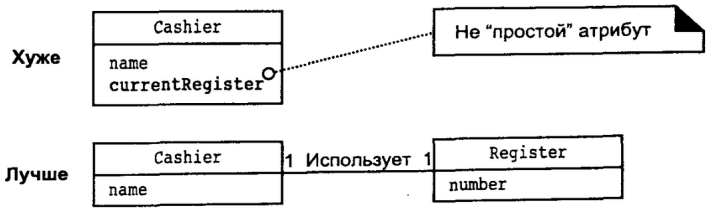


Рисунок 2.8 – Связь с помощью ассоциаций, а не атрибутов

Как видно из приведенного примера, стандартной ошибкой является моделирование сложного понятия предметной области в форме атрибута. Другими словами, аэропорт назначения на самом деле является не простой, а сложной сущностью с территорией, протянувшейся на многие километры. Таким образом, объект Flight (Полет) должен быть связан с объектом Airport (Аэропорт) с помощью ассоциации, а не атрибута (рис. 2.9).

Связывайте концептуальные классы с использованием ассоциаций, а не атрибутов.

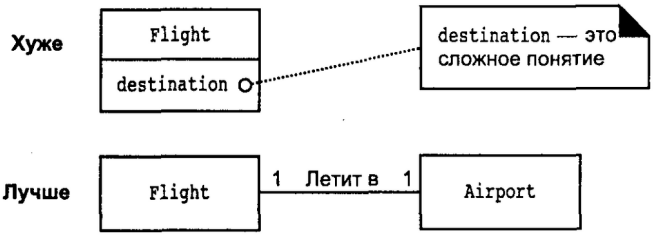


Рисунок 2.9 – Пример представления сложных понятий предметной области в виде ассоциации, а не атрибутов

Типы данных

Атрибутами должны быть данные простых типов (или в терминах UML *типов данных* – data types), для которых совершенно незначимой является уникальная тождественность (в контексте модели или системы).

Например, обычно не существует различия между:

- отдельными экземплярами числа 5 (тип Number);
- отдельными экземплярами строк 'cat' (тип String);
- отдельными объектами PhoneNumber, содержащими один и тот же номер телефона;
- отдельными объектами Address, содержащими один и тот же адрес.

Однако **существенными** являются различия между двумя отдельными объектами Person (Человек), даже если в обоих объектах содержится имя "Jill Smith", поскольку два экземпляра могут представлять отдельных людей с одним и тем же именем.

В терминах программных систем существует несколько случаев, когда нет необходимости сравнивать адреса памяти экземпляров объектов Number, String, PhoneNumber или Address, достаточно лишь выполнить сравнение их значений. Однако для того чтобы различить объекты Person, даже если они имеют одинаковые значения атрибутов, все же придется сравнить адреса памяти, поскольку в данном случае важна их уникальность.

Таким образом, все простые типы (числа, строки) считаются типами данных UML, однако не все типы данных являются простыми. Например, PhoneNumber (Номер телефона) – это не простой (или не примитивный) тип данных.

Данные простых типов называются также *объектами значений* (value objects).

Сущность данных простых типов трудноуловима. Для стандартной проверки "простоты" руководствуйтесь следующим **правилом**: если атрибут можно рассматривать как число, строку, логическое значение, дату или время (и т.д.), то следует оставить его в качестве атрибута; в противном случае его нужно представить отдельным концептуальным классом.

Если у вас имеются какие-либо сомнения, то лучше создайте отдельный концептуальный класс, а не атрибут.

Непримитивные типы классов

Тип атрибута может рассматриваться как непримитивный класс в его собственном значении в модели предметной области.

Например, в POS-системе имеется универсальный идентификатор товара. Обычно он рассматривается как простое число. Что же должно быть представлено как непримитивный класс? Руководствуйтесь следующими рекомендациями.

Тип данных, изначально считающийся примитивным (такой, как число или строка), может быть представлен в виде непримитивного класса в следующих случаях:

- Если он составлен из отдельных частей;
- Номер телефона, имя человека;
- Если с этим типом обычно ассоциируются операции, такие как синтаксический анализ и проверка;
- Номер страхового полиса;
- Он содержит другие атрибуты;
- Для льготной цены могут устанавливаться сроки действия (начало и конец);
- Если этот тип используется для задания количества с единицами измерения;
- Стоимость товара измеряется в некоторых единицах;
- Это абстракция одного или нескольких типов;
- Идентификатор товара – это некое обобщение типов UPC (Universal Product Code) и EAN (European Article Number).

Применение этих рекомендаций к атрибутам модели предметной области POS-системы приведет к формулировке следующих идей:

- *Идентификатор товара* – это абстракция, построенная на основе различных стандартных схем кодирования, включая UPC-A, UPC-E и семейство схем EAN. Эти схемы кодирования могут использоваться при подсчете контрольной суммы или иметь другие атрибуты (например, код изготовителя, товара, страны). Следовательно, это должен быть непримитивный класс ItemID.
- Атрибуты price (цена) и amount (сумма) должны иметь тип Quantity (Количество) или Money (Валюта), поскольку они представляют собой количество, выраженное в денежных единицах.

- Атрибут address (адрес) должен относиться к непримитивному типу Address, поскольку в нем имеются отдельные разделы.

Классы ItemID, Address и Quantity относятся к данным простых типов (уникальная тождественность является несущественной), так что их без проблем можно помещать в раздел атрибутов, а не связывать с использованием ассоциаций.

Совет разработчикам: не используйте атрибуты в качестве внешних ключей

Атрибуты не должны использоваться для связи концептуальных классов в модели предметной области. Наиболее распространенным нарушением этого принципа является добавление некоторой разновидности *атрибута внешнего ключа* (foreign key attribute), что обычно происходит при разработке реляционных баз данных. Например, на рис. 2.10 атрибут currentRegisterNumber использовать нежелательно, поскольку его назначением является связывание объекта Cashier с объектом Register. Объекты Cashier и Register лучше связать с помощью ассоциации, а не атрибута внешнего ключа. Не лишним будет повторить еще раз: связывайте типы с помощью ассоциаций, а не атрибутов.

Связать объекты можно множеством различных способов, и внешние ключи являются далеко не единственной возможностью. Для успешного создания системы на стадии проектирования необходимо решить, как реализовать требуемые связи.

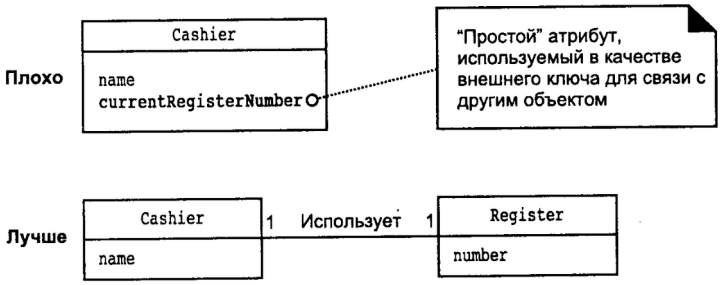


Рисунок 2.10 – Не используйте атрибуты в качестве внешних ключей

Моделирование атрибутов Quantity и Unit

Большинство количественных сущностей нельзя представить в виде числовых типов данных. Например, цена или скорость. С этими количественными сущностями связаны определенные единицы измерения, необходимые для поддержки функций конвертирования. Программная система ТТ предназначена для использования в различных странах, поэтому она должна поддерживать различные типы валют.

Решение заключается в создании отдельного концептуального класса **Quantity** (Количество), с которым будет ассоциироваться класс **Unit** (Единица измерения). Поскольку количество рассматривается в качестве типа данных, соответствующий атрибут можно поместить в раздел атрибутов, как показано на рис. 2.11. Обычно количество измеряют в некоторых единицах. Деньги – это количество, единицей измерения которого служит тип валюты. Вес – это количество, измеряемое в килограммах или фунтах.

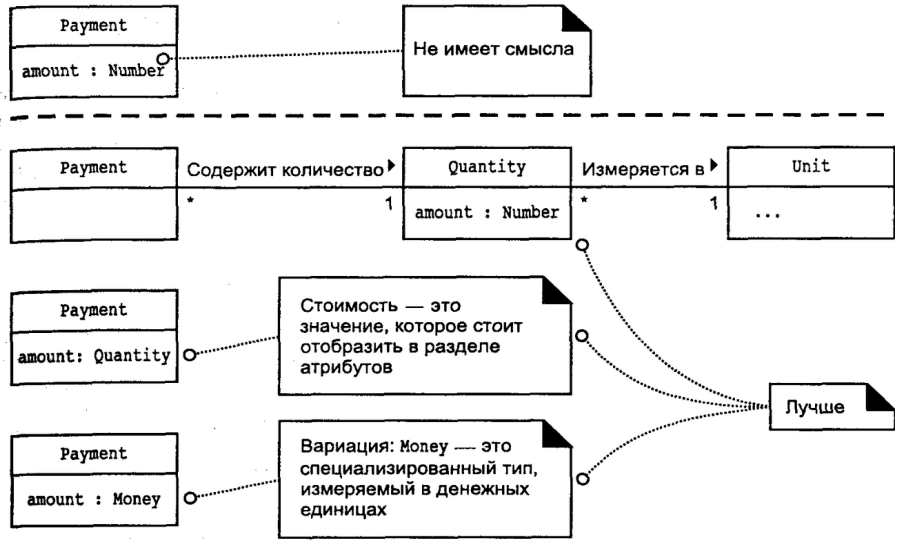


Рисунок 2.11 – Моделирование количества

Атрибуты модели предметной области системы ТТ

Теперь необходимо сформировать список атрибутов для отображения требований данной итерации или список атрибутов для сценариев прецедента Оформление продажи.

Таблица 2.2 – Список атрибутов для прецедента

Sale (Продажа)	**date (дата), time (время).** Товарный чек представляет собой распечатанные на бумаге данные о продаже. Как правило, на не также указываются дата и время продажи
Payment (Платеж)	**amount** (сумма). Если нужно определить, достаточную ли сумму заплатил покупатель, и вычислить разность между этой суммой и стоимостью покупок, необходимо иметь атрибут amount (известный также как "предложенная сумма")
Product Specification (Спецификация товара)	**description** (описание). Оказывается полезным, если описание товара нужно отобразить на экране или распечатать на товарном чеке **id** (идентификатор товара). Если необходимо просмотреть данные объекта ProductSpecification, соответствующего введенному значению кода **itemID**, то эту связь можно обеспечить с помощью данного атрибута **price** (цена). Необходим для вычисления итоговой суммы и отображения
SalesLineItem (Элемент продажи)	**quantity** (количество). Требуется для записи введенного количества товаров, если покупатель приобретает несколько единиц одного и того же товара (например, пять коробок конфет)
Store (Магазин)	**address** (адрес), **name** (название). На товарном чеке требуется указывать название и адрес магазина

Тогда модель предметной области с атрибутами будет иметь вид согласно рис. 2.13 и 2.14.

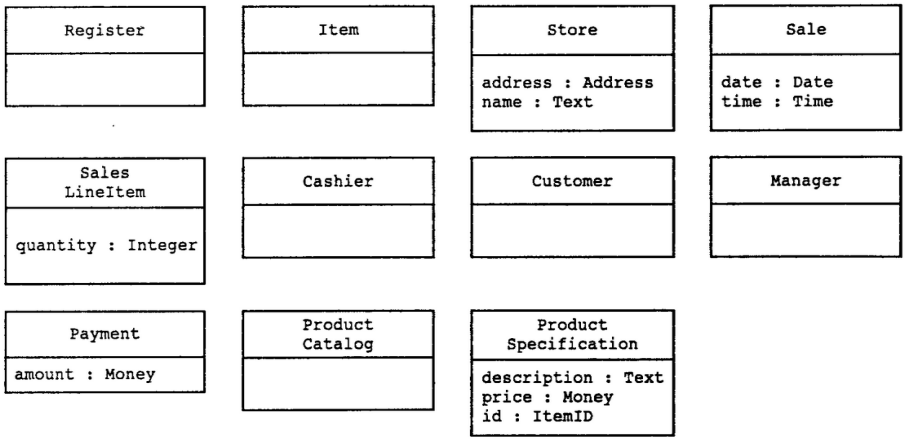


Рисунок 2.13 – Модель предметной области с атрибутами

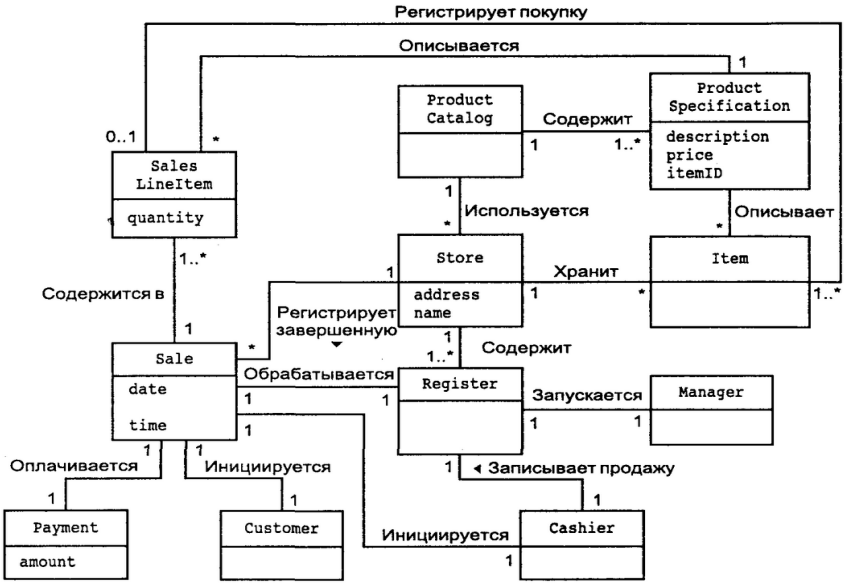


Рисунок 2.14 – Фрагмент модели предметной области

Задание на самостоятельную работу (для выбранной темы индивидуального проекта):

1. построить модель предметной области.