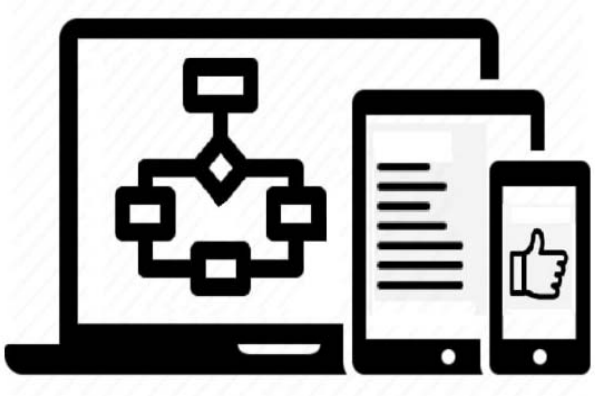
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Методические рекомендации к курсовому проектированию для студентов направлений подготовки 09.03.04 «Программная инженерия» и 09.03.01 «Информатика и вычислительная техника» дневной формы обучения



Могилев 2022

УДК 621.01
ББК 36.4
О87

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой ПОИТ «13» мая 2022 г., протокол № 9

Составители: канд. техн. наук, доц. Н. Н. Горбатенко;
ст. преподаватель О. В. Сергиенко;
ст. преподаватель Ю. В. Вайнилович

Рецензент ст. Ю. С. Романович

Методические рекомендации разработаны на основе рабочей программы по дисциплине «Объектно-ориентированное программирование и проектирование» для студентов направлений подготовки 09.03.04 «Программная инженерия» и 09.03.01 «Информатика и вычислительная техника» и предназначены для использования при выполнении курсового проекта.

Учебно-методическое издание

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Ответственный за выпуск	В. В. Кутузов
Корректор	Т. А. Рыжикова
Компьютерная верстка	М. М. Дударева

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

1 Общие положения

1.1 Цель курсового проекта

Целями курсового проектирования являются:

– закрепление знаний, полученных в ходе теоретического и практического изучения дисциплины «Объектно-ориентированное программирование»;

– приобретение навыков практического программирования с использованием объектно-ориентированной парадигмы;

– изучение современных систем программирования и сред для разработки объектно-ориентированных программ и систем;

– изучение отдельных разделов предметной области, не вошедших в программу теоретического обучения, формирование навыка поиска информации по конкретной теме, ее анализа и использования для решения задачи;

– подготовка к выполнению выпускной квалификационной работы.

Курсовой проект позволяет сформировать способности будущего специалиста к самостоятельному решению практических задач и инженерных проблем с использованием теоретических положений, а также знаний и умений, полученных в ходе обучения объектно-ориентированному программированию.

Задачей курсового проектирования является разработка иерархии классов в заданной предметной области, включающая в себя:

– разработку модели классов на языке UML;

– разработку диаграмм взаимодействия на языке UML;

– разработку абстрактных классов и интерфейсов;

– генерацию кода классов;

– разработку приложения с использованием созданной иерархии классов;

– разработку документации.

1.2 Тематика курсового проектирования

В курсовом проекте необходимо построить иерархию классов для заданной предметной области и создать приложение на основе этой иерархии. Примерная тематика курсовых проектов:

– разработка информационной системы «Автоматическая телефонная станция»;

– разработка информационной системы «Почтовая служба»;

– разработка информационной системы «Магазин музыкальных инструментов»;

– исследовательские проекты; они включают в себя разработку различных программных компонентов, связанных с языками программирования и средами разработки. Примерами могут служить интерпретация языка UML, средства визуального программирования и т. п.

В случаях, когда сложность проекта достаточно велика и (или) необходима подробная разработка тем, кафедра имеет право выдавать комплексные задания на курсовое проектирование группе студентов. Такими заданиями могут быть

проекты по созданию перспективных разработок в области языков программирования и сред разработки программного обеспечения.

Для проектирования диаграмм классов могут быть использованы как специализированные программные средства, так и средства, встроенные в современные системы программирования и проектирования.

В качестве языка программирования в курсовом проекте допускается использование различных объектно-ориентированных языков, таких как C#, Java, C++ и др.

1.3 Руководство проектом

Общее методическое руководство курсовым проектом выполняет его руководитель. Руководитель курсового проекта обязан:

– выдать задание на курсовое проектирование;

– оказывать консультационную помощь в подготовке плана курсового проекта, графика его выполнения, в подборе литературы и фактического материала;

– содействовать студенту в выборе методики реализации проекта;

– осуществлять систематический контроль за ходом выполнения курсового проекта в соответствии с планом и графиком его выполнения;

– информировать деканат о случаях несоблюдения студентом графика выполнения курсового проекта;

– давать студенту квалифицированные рекомендации по содержанию курсового проекта;

– производить оценку качества выполнения курсового проекта в соответствии с предъявляемыми к ней требованиями.

Руководитель курсового проекта имеет право:

– выбрать удобную для него и студента форму организации взаимодействия, в том числе согласовать график подготовки курсового проекта и установить периодичность личных встреч или иных контактов;

– по результатам каждой встречи требовать, чтобы студент подготовил и согласовал с ним краткое резюме полученных рекомендаций и намеченных дальнейших шагов по подготовке курсового проекта;

– требовать, чтобы студент внимательно относился к полученным рекомендациям и являлся на встречи подготовленным;

– при выставлении оценки принять во внимание соблюдение студентом контрольных сроков графика подготовки курсового проекта.

2 Структура и содержание курсового проекта

Курсовой проект включает в себя демонстрационную версию приложения, пояснительную записку и графическую часть.

Демонстрационная версия приложения – это программа, осуществляющая решение задачи, сформулированной в задании на курсовое проектирование.

Пояснительная записка представляет собой текстовый документ, в котором, используя общепринятую научно-техническую терминологию, сжато, логично и аргументированно излагают порядок выполнения всех этапов курсового проекта, описывают достигнутые результаты, раскрывают и аргументируют принятые решения, обосновывают их целесообразность.

Графическая часть включает различные схемы, графики, диаграммы, созданные в ходе выполнения проекта. Это могут быть: диаграммы классов; диаграммы деятельности; диаграммы последовательности; скриншоты экранных форм программы; схемы алгоритмов и т. п.

Рекомендуемый объем пояснительной записки от 25 до 35 страниц печатного текста формата А4 (без учета приложений), графической части – один лист формата А1.

Пояснительная записка должна содержать следующие элементы.

Титульный лист.

Бланк задания на курсовой проект.

Содержание.

Введение.

1 Постановка задачи.

2 Анализ предметной области.

2.1 Выделение классов и ассоциаций.

2.2 Выделение атрибутов классов.

2.3 Выделение наследования.

2.4 Модель состояния классов.

3 Анализ приложения.

3.1 Модель взаимодействия приложения.

3.2 Модель классов приложения.

3.3 Модель состояния приложения.

4 Проектирование классов.

4.1 Реализация вариантов использования с помощью операций.

4.2 Разработка алгоритмов операций.

5 Тестирование приложения.

6 Описание работы приложения.

Заключение.

Список литературы.

Приложение.

Титульный лист является первой страницей пояснительной записки, но номер страницы на нем не ставится. Образец оформления титульного листа представлен в приложении А.

Бланк задания на курсовой проект выдает руководитель курсового проекта. В нём указывается тема проекта, исходные данные, содержание пояснительной записки и графической части проекта, календарный график работы над проектом.

Содержание включает порядковые номера и наименование структурных элементов пояснительной записки с указанием номера страницы, на которой они помещены.

Введение. Во введении раскрывается цель и задачи проекта, приводится краткое описание содержания последующей основной части пояснительной записки.

Цель проекта, это то, к чему стремится автор в своей работе, т. е. конечный результат проектирования. Цель проекта должна быть созвучна названию темы проекта, например, целью проекта является разработка иерархии классов информационной системы «Автоматическая телефонная станция».

После формулировки цели проекта следует указать конкретные задачи, которые предстоит решать для достижения этой цели. Это обычно делается в форме перечисления (изучить, описать, установить, выявить, вывести формулу, разработать и т. п.). Формулируя задачи, следует учитывать, что описание их решения должно составить содержание разделов курсового проекта.

Постановка задачи. В разделе описывается содержание задачи, указанной в задании на курсовое проектирование.

Анализ предметной области. Описывается исследование предметной области, выявляются классы, ассоциации классов, атрибуты классов, наследование классов, составляется модель состояния предметной области.

Анализ приложения. Осуществляется идентификация лиц взаимодействующих с приложением, строится диаграмма вариантов использования, описываются варианты сценариев взаимодействия пользователей с системой, разрабатываются классы приложения и модель состояния приложения.

Проектирование классов. Осуществляется реализация вариантов использования с помощью операций и разрабатываются алгоритмы операций классов предметной области и приложения.

Тестирование приложения. Составляется набор тестовых примеров, обеспечивающих проверку работоспособности всех методов классов. Приводятся и анализируются результаты тестирования.

Описание работы приложения. Приводится описание работы приложения, созданного в ходе курсового проектирования.

Заключение должно содержать выводы по выполнению задания на проект и соответствовать введению в смысле достижения указанных в нем поставленной цели и задач проектирования. Следует отметить преимущества, связанные с реализацией проектных предложений, отметить недостатки работы, дать практические рекомендации по совершенствованию объекта проектирования, охарактеризовать перспективы дальнейшего развития работы. Рекомендуемый объем раздела – одна страница.

Список литературы – это упорядоченный в алфавитно-хронологической последовательности перечень библиографических описаний документальных источников информации по теме курсового проекта.

Приложение. Приводится распечатка текста приложения, разработанного в ходе выполнения курсового проекта.

3 Выполнение курсового проекта

3.1 Исходные данные

1 Тема курсового проекта выбирается из списка, предлагаемого руководителем проекта.

2 Язык программирования – C#, C++, Java.

3 Среда разработки – Visual Studio 2019 и выше.

4 Вид приложения – консольное или Windows Forms приложение.

5 Парадигма программирования – объектно-ориентированная.

6 Способ организации данных – массивы, кортежи, коллекции, структуры, классы.

7 Способ хранения данных – в виде отдельных файлов на диске компьютера (файловая база данных).

3.2 Планирование времени

Перед началом проектирования следует проанализировать, из каких этапов состоит проект, оценить сложность этапов и правильно распределить время, отводимое для выполнения курсовой работы, между отдельными этапами.

Рекомендуемое распределение времени на выполнение курсового проекта приведено в таблице 3.1.

Таблица 3.1 – Календарный график выполнения курсового проекта

Месяц	Неделя месяца и объем	Содержание этапа выполнения курсового проекта	Представляемый результат
Февраль	1, 2	Выдача задания на курсовое проектирование	
	3, 4 (10 %)	Постановка задачи Анализ предметной области	Модель предметной области
Март	1, 2, 3, 4 (40 %)	Анализ приложения	Модель приложения
Апрель	1, 2, 3, 4 (70 %)	Проектирование классов	Рабочая версия программного продукта
Май	1, 2 (80 %)	Тестирование приложения	Результаты тестирования программного продукта
	3, 4 (95 %)	Оформление пояснительной записки и графической части проекта. Сдача курсового проекта на проверку	Демонстрационная версия программного продукта. Полностью оформленная пояснительная записка и графическая часть проекта
Июнь	1 (100 %)	Подготовка к защите и защита курсового проекта	Доклад (5…7 мин)

3.3 Постановка задачи

На данном этапе осуществляется сбор информации по теме проекта и составляется краткое описание задания на разработку программного продукта. В этом описании следует отразить:

– назначение создаваемого приложения;

– область применения приложения;

– список задач, решаемых приложением;

– входные данные и результаты работы приложения.

Формулировка задачи может иметь разную степень детализации, быть неполной или неформальной. Последующий анализ делает ее более точной и выявляет неоднозначности и несогласованности. Формулировку задачи не следует воспринимать как нечто неизменное. Она должна служить основой для определения реальных требований к приложению. Разработчик должен достичь понимания реальной системы, которую описывает формулировка задачи, и представить ее важнейшие черты в виде объектной модели предметной области и объектной модели приложения.

Образцы формулировки задания на разработку программного продукта приведены в приложениях Б и В.

3.4 Анализ предметной области

Целью анализа является формирование модели классов предметной области. Модель классов отражает статическую структуру системы и делит ее на отдельные элементы, удобные для использования. Модель предметной области строится на основании информации, получаемой из постановки задачи, при изучении артефактов в аналогичных и связанных системах, из знаний экспертов в области приложения, а также из общих знаний о реальном мире. Необходимо учитывать всю доступную информацию и не полагаться на сведения, полученные из одного источника.

3.4.1 Выделение классов и ассоциаций. Начинать работу нужно с перечисления всех потенциальных классов из письменного описания задачи. Классы часто соответствуют существительным. Например, из предложения «система бронирования билетов на представления в различных театрах» можно выделить потенциальные классы «бронирование», «система», «билет», «представление» и «театр». Однако не следует переписывать все подряд. Идея состоит в том, чтобы отразить в модели понятия. Не все существительные описывают понятия, и наоборот: понятия могут выражаться другими частями речи.

Пример с банкоматом. В результате выделения понятий из описания задачи о банкомате (см. приложение Б), можно получить потенциальные классы, перечисленные на рисунке 3.1.

Рисунок 3.1 – Потенциальные классы для модели банкомата, полученные из знаний о предметной области

Теперь нужно отбросить ненужные и некорректные классы, используя перечисленные ниже критерии. Рисунок 3.2 показывает, какие классы были удалены из модели банкомата.

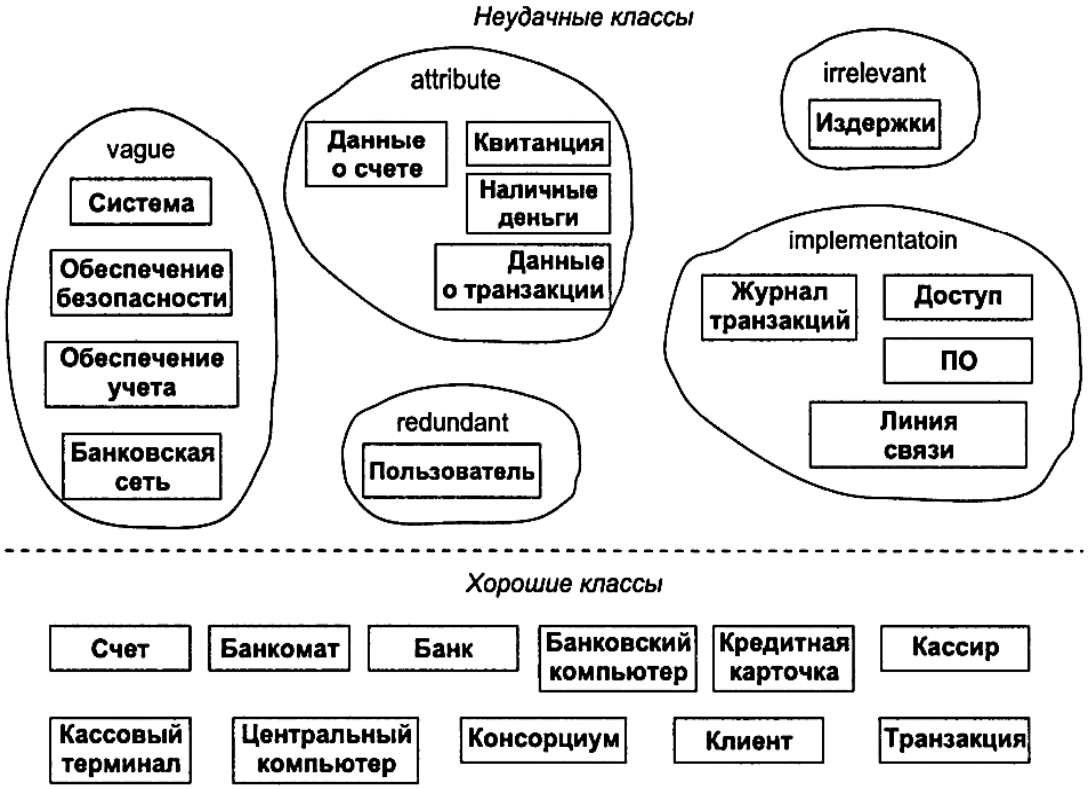


Рисунок 3.2 – Удаление лишних классов из модели банкомата

Избыточные классы. Если два класса выражают одно и то же понятие, нужно оставить тот, название которого лучше описывает сущность понятия. В примере с банкоматом Customer (Клиент) и User (Пользователь) – избыточные классы; мы сохраняем класс Customer как более соответствующий сути понятия.

Несущественные классы. Если класс имеет слабое отношение к задаче, выбросьте его. Это решение принимается субъективно, потому что важность класса зависит от контекста.

Пример с банкоматом. Класс Cost (Стоимость) лежит вне области классов, относящихся к программному обеспечению банкомата.

Нечеткие классы. Класс должен быть четко определен. Некоторые потенциальные классы могут иметь нечетко определенные границы или слишком широкую область охвата.

Пример с банкоматом. Класс RecordKeepingProvision (Средство для ведения записей) определен нечетко, а его задачи решаются классом Transaction (Транзанкция).

Операции. Если название описывает операцию, которая применяется к объектам, и она не рассматривается сама по себе, его следует исключить из списка классов. Например, телефонный звонок – это последовательность действий, в которых участвует звонящий и телефонная сеть.

Конструкции, *относящиеся к реализации*. Модель не должна содержать конструкций, не принадлежащих к реальному миру. Они могут потребоваться на этапе проектирования, но не этапе анализа.

Пример с банкоматом. Такие сущности, как процессор, алгоритм, прерывание и подпрограмма, являются конструкциями, относящимися к реализации. В примере с банкоматом класс TransactionLog (Журнал Транзанкций) относится к реализации.

Производные классы. Как правило классы, которые могут быть выведены из других классов, не следует включать в модель.

На следующем этапе следует выделить ассоциации между классами. Структурное отношение между двумя и более классами является ассоциацией. Если один класс ссылается на другой класс, это тоже ассоциация.

Ассоциации часто соответствуют глаголам состояния или глагольным группам. К ним относятся характеристики физического размещения (NextTo – РядомС, PartOf – Часть, ContainnedIn – СодержитсяВ), направленные действия (Drives – Управляет), передача информации (TolksTo – РазговариваетС), владение (Has – Имеет, PartOf – Часть) и выполнение некоторого условия (WorksFor – РаботаетНа, MarriedTo – ЖенатНа, Manages – Управляет). Сначала выделите все потенциальные ассоциации из описания задачи и запишите их на бумаге. Суть в том, чтобы выделить все отношения, в какой бы форме они не были выражены на естественном языке.

На рисунке 3.3 показаны ассоциации для примера с банкоматом. Большинство из них соответствуют глагольным формам из описания задачи. Некоторые ассоциации присутствовали в описании задачи неявным образом. Другие были введены в модель на основании знаний о реальном мире или предположений о сути задачи.

Теперь можно отбросить ненужные и некорректные ассоциации, руководствуясь перечисленными ниже критериями.

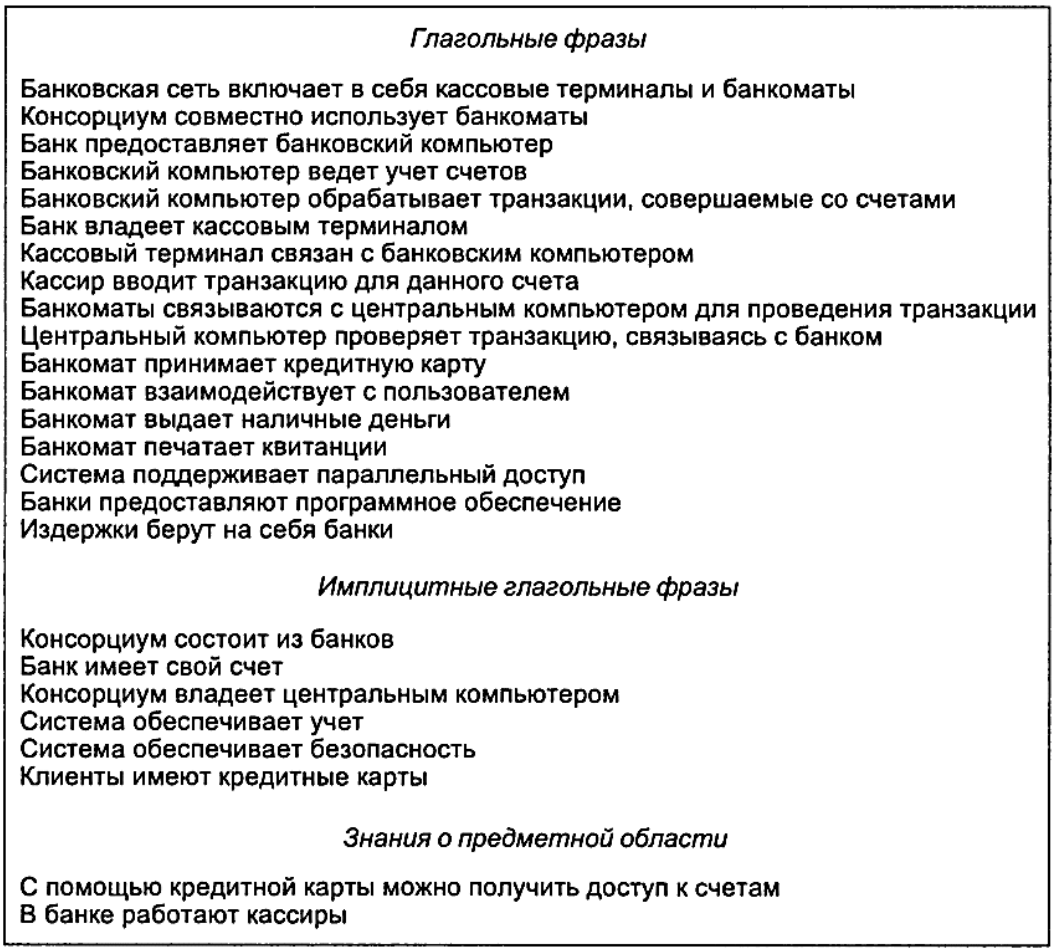


Рисунок 3.3 – Ассоциации, выделенные из описания задачи о банкомате

Ассоциации между классами, которые были удалены на предыдущих этапах. Если один из классов, связываемых ассоциацией, был исключен, ассоциацию тоже нужно исключить или переформулировать в терминах других классов.

Пример с банкоматом. Мы можем исключить ассоциации Banking network includes cashier stations and ATMs (Банковская сеть включает в себя кассовые терминалы и банкоматы), ATM dispenses cash (Банкомат выдает наличные), ATM prints receipts (Банкомат печатает чеки), Banks provide software (Банки предоставляют программное обеспечение), Cost apportioned to banks (Стоимость распределяется между банками), System provides recordkeeping (Система обеспечивает ведение записей), System provides security (Система обеспечивает безопасность).

Несущественные, или относящиеся к реализации ассоциации. Выбросьте ассоциации, лежащие за пределами области задачи или описывающие конструкции, относящиеся к реализации.

Пример с банкоматом. Ассоциация System handles concurrent access (Система обрабатывает параллельные обращения) описывает понятие, относящиеся к реализации.

Действия. Ассоциация должна описывать структурное свойство области приложения, а не кратковременное событие.

Пример с банкоматом. Ассоциация ATM accept cash card (Банкомат принимает банковскую карту) описывает часть цикла взаимодействия банкомата с клиентом, а не постоянное отношение банкомата и карты. По этой же причине можно исключить ассоциацию ATM interacts whih user (Банкомат взаимодействует с пользователем).

Тернарные ассоциации. Большинство *n*-арных ассоциаций можно выразить через бинарные, добавив соответствующие квалификаторы.

Пример с банкоматом. Ассоциацию Bank computer processer transaction against account (Банковский компьютер обрабатывает транзанкцию по счету) можно разбить на Bank computer processes transaction (Банковский компьютер обрабатывает транзанкцию) и Transaction concerns account (Транзанкция связана со счетом).

Производные ассоциации. Отбросьте ассоциации, которые могут быть выражены через другие ассоциации. Классы, атрибуты и ассоциации модели классов должны отражать максимально независимую информацию. Наличие множества маршрутов между классами часто указывает на производные ассоциации, которые могут быть выражены через примитивные ассоциации. Consortium shares ATMs (Консорциум совместно владеет банкоматами) – это композиция ассоциаций Consortium owns central computer (Консорциум владеет центральным компьютером) и Cenrral computer communicates with ATMs (Центральный компьютер взвимодействует с банкоматами).

Дальше следует проанализировать семантику ассоциаций.

Неправильно названные ассоциации. Название должно говорить не о том, как или почему возникла некая ситуация, а о том, в чем эта ситуация состоит. Названия важны для понимания модели в целом, а потому выбирать их следует очень осторожно.

Пример с банкоматом. Bank computer maintains accounts (Банковский компьютер обслуживает счета) – это утверждение, описывающее действие. Название ассоциации правильнее будет переформулировать как Bank holds account (Банк управляет счетом).

Названия полюсов ассоциаций. Названия полюсов можно указывать везде, где они имеют смысл. Например, в ассоциации WorksFor (РаботаетНа) класс Company (Компания) по отношению к классу Person (Человек) является работодателем, а Person по отношению к классу Company представляет собой сотрудника (employee). Если пару классов связывает только одна ассоциация и ее смысл очевиден, названия полюсов можно не указывать.

Квалифицированные ассоциации. Обычно название идентифицирует объект в рамках некоторого контекста. Большинство названий не являются уникальными в масштабах всей системы (глобально). Контекст вместе с именем позволяют уникально идентифицировать объект.

Пример с банкоматом. Квалификатор bankCode (Код банка) позволяет отличать друг от друга банки, входящие в состав консорциума. Каждая карта должна иметь банковский код, чтобы транзакции можно было направлять в выдавший ее банк.

Кратность. Этот параметр ассоциаций указывать нужно, но не старайтесь определить его точно на первом этапе моделирования. Кратность часто изменяется в процессе анализа.

Агрегация. Агрегация важна для некоторых видов приложений, в частности для описания деталей механизмов и спецификаций материалов. Для других приложений важность не столь велика, и не всегда бывает понятно, следует ли ее использовать вместо обычной ассоциации. Не тратьте слишком много времени на определение типа ассоциации. Выберите то, что сразу покажется более правильным, и двигайтесь дальше.

Пример с банкоматом. Мы считаем Bank (Банк) частью Consortium (Консорциума) и обозначаем отношение между ними как агрегацию.

На рисунке 3.4 приведена диаграмма классов с нанесенными на нее ассоциациями. На ней указаны имена только наиболее важных ассоциаций.

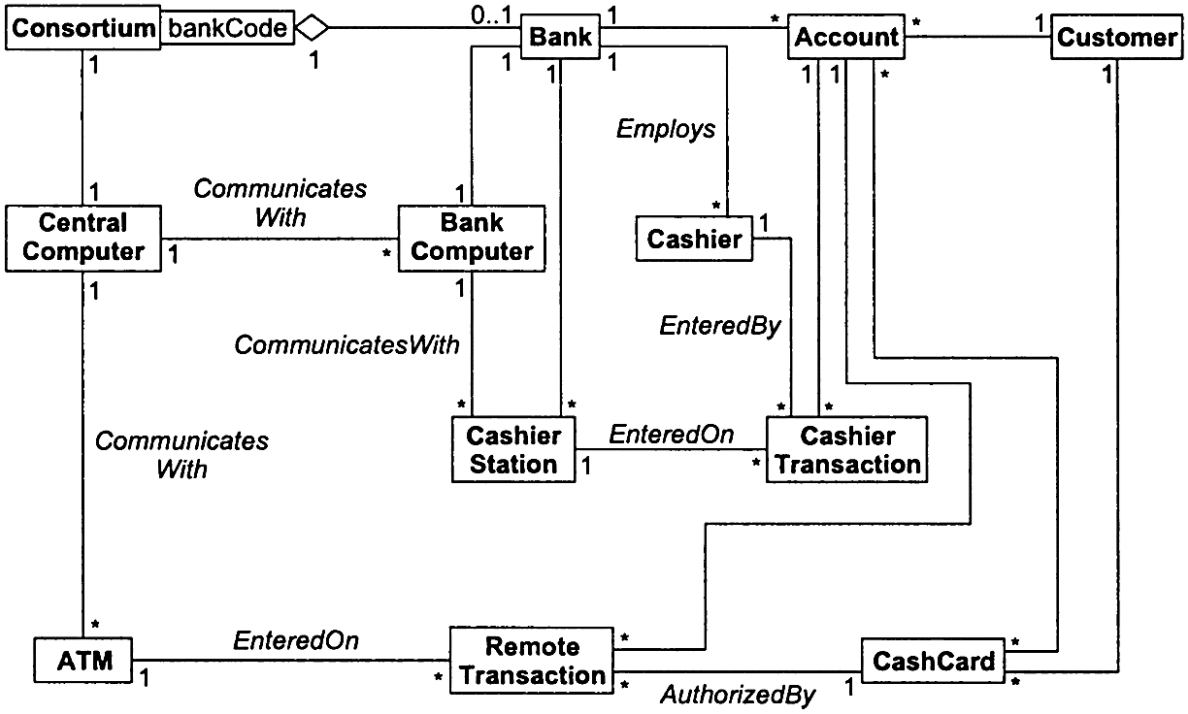


Рисунок 3.4 – Исходная диаграмма классов для банкомата

3.4.2 Выделение атрибутов классов. Атрибуты – это свойства объектов, такие как вес, скорость или цвет. Значения атрибутов не должны быть объектами. Атрибуты обычно присутствуют в описании задачи в виде существительных, участвующих в притяжательных оборотах, наподобие «цвет машины» или «положение курсора». Прилагательное часто соответствует конкретным значениям атрибутов-перечислений (например, «красный», «включен»). В отличие от классов и ассоциаций атрибуты вряд ли будут полностью перечислены в описании задачи. Вам придется опираться на свое знание области задачи и реального мира, чтобы выделить их все. Атрибуты также присутствуют в артефактах родственных систем.

Ищите атрибуты и для ассоциаций. Эти атрибуты являются свойствами связей между объектами, а не свойствами индивидуальных объектов. Например, ассоциация «многие – ко – многим» между StockHolder (Держатель акций) Company (Компания) имеет атрибут numberOfShares (количество акций).

Лишние атрибуты следует исключить, руководствуясь следующими критериями.

Объекты. Если важной чертой элемента является независимое существование, то этот элемент – объект (а не атрибут). Например, босс – это класс, а зарплата – атрибут. Разница часто зависит от приложения. Например, в Списке рассылки Город может быть атрибутом, тогда как в Переписи населения Город будет классом с множеством атрибутов. Элемент, обладающий собственными чертами в рамках данного приложения, должен моделироваться как класс.

Квалификаторы. Если значение атрибута зависит от конкретного контекста, его можно попробовать переформулировать в виде квалификатора. Например, номер Сотрудника – это не уникальная характеристика человека, занятого на двух работах, это, скорее, квалификатор ассоциации «Компания нанимает сотрудника».

Имена. Имена лучше моделировать как квалификаторы, а не атрибуты. Ответьте на следующие вопросы: служит ли имя для выбора уникального объекта из множества; может ли объект, принадлежащий множеству, иметь более одного имени?

Если ответ положительный, имя служит квалификатором ассоциации. Если имя кажется уникальным, возможно, Вы забыли добавить в модель класс, для которого оно служит квалификатором.

Имя является атрибутом в том случае, если его использование не зависит от контекста, в особенности если оно неуникально в некотором множестве. Имена людей в отличие от названия компаний могут повторятся, а потому являются атрибутами.

Идентификаторы. Объектно-ориентированные языки программирования используют понятие идентификатора объекта для обозначения однозначной ссылки на объект. Не следует включать в модель атрибут, единственным назначением которого является идентификация, потому что идентификаторы присутствуют в моделях классов явным образом. Перечисляйте только те атрибуты, которые присутствуют в области приложения.

Атрибуты ассоциаций. Если существование значения требует существования связи, соответствующее свойство является атрибутом ассоциации, а не одного из связанных ею классов. Атрибуты обычно достаточно ясно выделяются из ассоциаций «многие – ко – многим»: их нельзя прикрепить ни к одному из классов из-за их кратности. Например, атрибут дата Вступления» принадлежит ассоциации между Человеком и Клубом, потому что человек может принадлежать к нескольким клубам, а клуб может иметь несколько членов. Ассоциации «один – ко – многим» детализировать сложнее, потому что тут можно прикрепить атрибут к одному из классов без потери информации. Сопротивляйтесь этому желанию, потому что если кратность изменится, модель станет некорректной. Те же проблемы возникают и с ассоциациями типа «один – к – одному».

Внутренние значения. Если атрибут описывает внутреннее состояние объекта, невидимое снаружи, его следует исключить из модели.

Нетипичные атрибуты. Атрибут, полностью отличающийся от всех остальных и не связанный с ними, может указывать на то, что класс, к которому он относится, следует разделить на два разных класса. Класс должен быть простым и цельным.

Логические атрибуты. Рассмотрите логические атрибуты еще раз. Часто логический атрибут можно расширить и переформулировать в виде перечисления.

Пример с банкоматом. Применим эти критерии к примеру, с банкоматом, чтобы получить список атрибутов для каждого класса. Некоторые потенциальные атрибуты стали квалификаторами ассоциаций. Рассмотрим следующие аспекты модели:

– bankCode (код Банка) и cardCode (код Карты) хранятся на карте. Их формат зависит от реализации, но мы обязаны, по крайней мере, создать ассоциацию Bank issue CashCard (Банк выдает Кредитную Карту). Код карты становится квалификатором этой ассоциации, а код банка – квалификатором банка по отношению к консорциуму;

– состояние компьютеров не имеет отношения к нашей задаче. Включенное или выключенное состояние компьютера – это атрибут, являющийся частью реализации;

– класс Consortium обеспечивает контекст для квалификатора bankCode (код Банка) и может быть полезен для дальнейшего расширения.

На рисунке 3.5 показана модель классов банкомата после добавления атрибутов.

3.4.3 Выделение наследования. Следующий шаг: организация классов при помощи наследования путем выявления их общей структуры. Наследование может быть использовано двумя способами: как обобщение одинаковых аспектов существующих классов в суперклассах (снизу вверх) и как конкретизация существующих классов множеством подклассов (сверху вниз).

Обобщение снизу вверх. Наследование можно прослеживать снизу вверх путем поиска классов с одинаковыми атрибутами, ассоциациями и операциями. Для каждого обобщения следует определить суперкласс, в котором будут содержаться общие черты. Для этого, возможно, придется несколько переопределить некоторые атрибуты или классы.

Пример с банкоматом. Классы RemoteTransaction (Удаленная Транзакция) и CashierTransaction (Кассовая Транзакция) подобны друг другу (за исключением инициализации) и могут быть обобщены классом Transaction (Транзакция).

Конкретизация сверху вниз. Конкретизация обычно следует из описания области приложения. Ищите именные группы, состоящие из различных прилагательных с именем класса: флуоресцентная лампа, лампа накаливания, раскрывающееся меню, выдвигающееся меню. Если предлагаемая конкретизация несовместима с существующим классом, возможно, этот класс просто неправильно сформулирован.

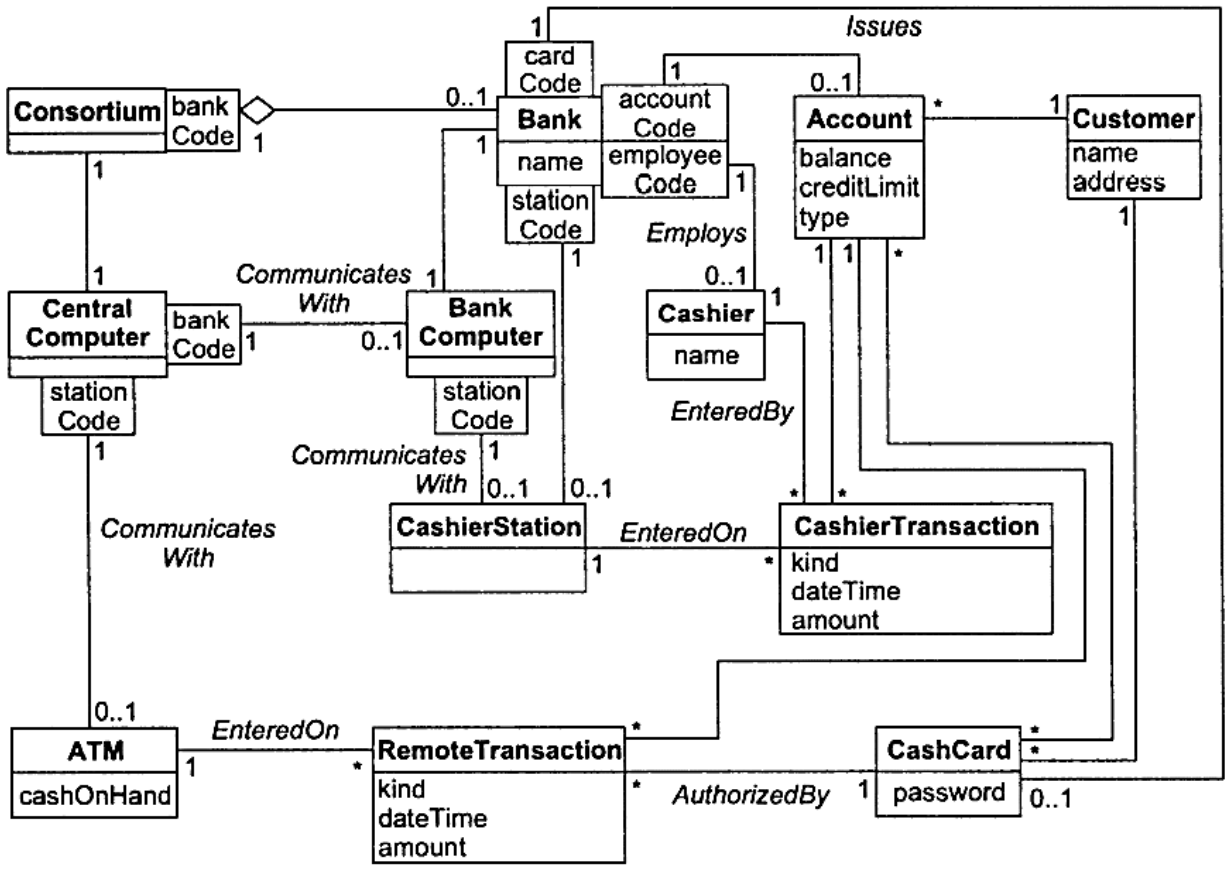


Рисунок 3.5 – Модель классов банкомата с атрибутами

Множественное наследование. Вы можете использовать множественное наследование, но только тогда, когда это действительно необходимо, т. к. оно повышает концептуальную и техническую сложность модели.

Подобные ассоциации. Если название ассоциации появляется в модели несколько раз, причем несет оно при этом одинаковый смысл, попробуйте обобщить ассоциируемые классы. Иногда у таких классов может не быть ничего общего за исключением ассоциации, но достаточно часто вы будете обнаруживать общие черты, пропущенные на предыдущих этапах.

Пример с банкоматом. Transaction (Транзакция) вводится как посредством CashierStation (Касса), так и на ATM (Банкомат). Класс EntryStation (Устройство Ввода) обобщает классы CashierStation и ATM.

Корректирование уровня наследования. Атрибуты и ассоциации должны быть присвоены конкретным классам из иерархии. Каждый из них должен быть присвоен наиболее общему классу, к которому он применим. Для этого могут потребоваться некоторые корректировки. Из симметрии может следовать наличие дополнительных атрибутов, которые позволят более четко отличать подклассы друг от друга.

На рисунке 3.6 показана модель классов банкомата после добавления наследования.

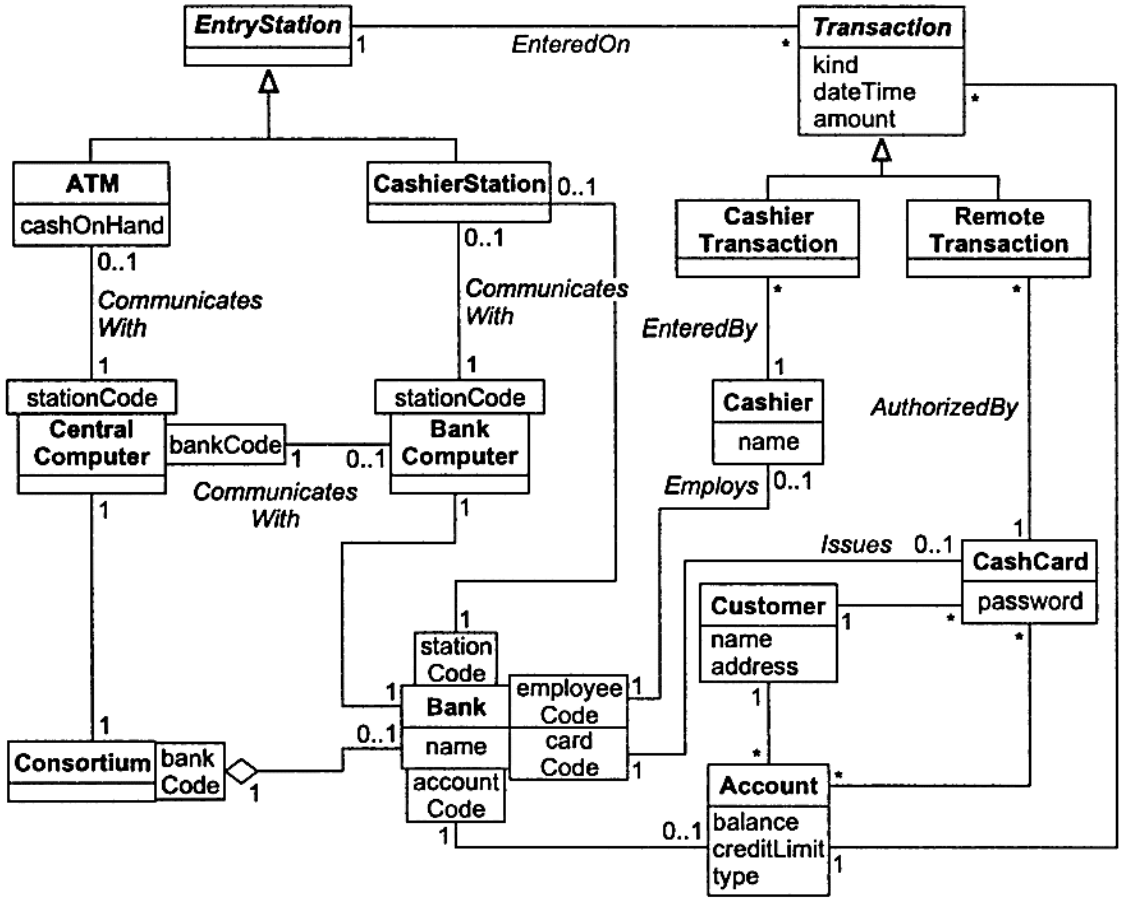


Рисунок 3.6 – Модель классов банкомата после добавления наследования

3.4.4 Модель состояния предметной области. Некоторые объекты предметной области за время своей жизни сменяют несколько качественно различных состояний. В этих состояниях они могут иметь разные ограничения на значения атрибутов, разные ассоциации или кратности, выполнять разные операции и т. д. Часто бывает полезно построить диаграмму состояний для таких классов. Диаграмма состояний описывает различные состояния, в которых может находится объект, свойства объекта, а также события, вызывающие переход объекта из одного состояния в другое.

Большинство классов предметной области не требуют использования диаграмм состояний. Для их понимания достаточно списка операций. Модель состояний может помочь в понимании поведения тех классов, которые могут находится в существенно различных состояниях.

Модель состояний предметной области создается в несколько этапов.

Выявление классов с разными состояниями. Изучите список классов предметной области, характеризующихся четко определенным жизненным циклом. Ищите классы, которые развиваются или имеют циклическое поведение. Идентифицируйте значимые состояния в жизненном цикле каждого из объектов. Например, научная статья для некоторого журнала последовательно переходит из состояния «Написание» в состояние «Рецензирование», а затем в состояние «Принята» или «Отвергнута». Состояние статьи могут сменяться циклично, например, если рецензенты потребуют внесения изменений или дополнений,

но основная последовательность состояний может быть охарактеризована как развитие.

Пример с банкоматом. Account (Счет) – это важное понятие из области бизнеса. Поведение банкомата зависит от состояния счета. Жизненный цикл счета – смесь последовательного и циклического поведения. Другие классы предметной области банкомата не имеют значимых моделей состояний предметной области.

Выделение состояний. Перечислите состояния для каждого из классов. Охарактеризуйте объекты каждого класса: укажите значения атрибутов, которые может имеет объект, ассоциации, в которых он может участвовать, значения кратности узлов этих ассоциаций, определите атрибуты и ассоциации, которые имеют смысл только в определенных состояниях. Дайте каждому состоянию осмысленное название.

Пример с банкоматом. Класс Account (Счет) может находиться в состояниях Normal (нормальное – обычный режим доступа), Closed (закрытый – клиент закрыл свой счет, но он еще не был исключен из записей банка), Overdrawn (с превышенным кредитным лимитом – клиент превысил кредитный лимит по своему счету) и Supended (приостановлен – доступ к счету был заблокирован по каким-либо причинам).

Выделение событий. Получив предварительный список состояний, займитесь поиском событий, которые вызывают переходы между этими состояниями. Подумайте о внешних воздействиях, которые вызывают изменения состояний. Во многих ситуациях событие можно рассматривать как завершение текущей деятельности. Например, если статья находится в состоянии «Рецензирование», переход из этого состояния осуществляется по завершении деятельности рецензента. Решение может быть положительным (переход в состояние «Принята») или отрицательным (переход в состояние «Отвергнута»).

Другие состояния можно обнаружить, поразмыслив о том, каким образом объект может попасть в определенное состояние. Например, если вы снимаете трубку телефона, он переходит в состояние «Набор номера».

Пример с банкоматом. Перечислим важнейшие события в нашей модели: close account (закрытие счета), withdraw excess funds (превышение кредитного лимита), repeated incorrect PIN (повторный неправильный ввод PIN-кода), suspected fraud (возможная подделка) и administrative action (административные действия).

Построение диаграмм состояний. Распределите события по состояниям, к которым они относятся. Добавьте переходы, показывающие изменения состояний, вызванные осуществлением события в то время, когда объект находится в определенном состоянии. Если событие завершает состояние, из этого состояния обычно бывает только один переход в другое состояние. Если событие инициирует целевое состояние, вы должны рассмотреть, в каких состояниях это событие может происходить, и добавить на диаграмму переходы из этих состояний в целевое состояние. Если событие имеет разное действие на разные состояния, добавьте переходы для каждого из этих состояний.

Пример с банкоматом. На рисунке 3.7 показана модель состояний предметной области для класса Account (Счет).

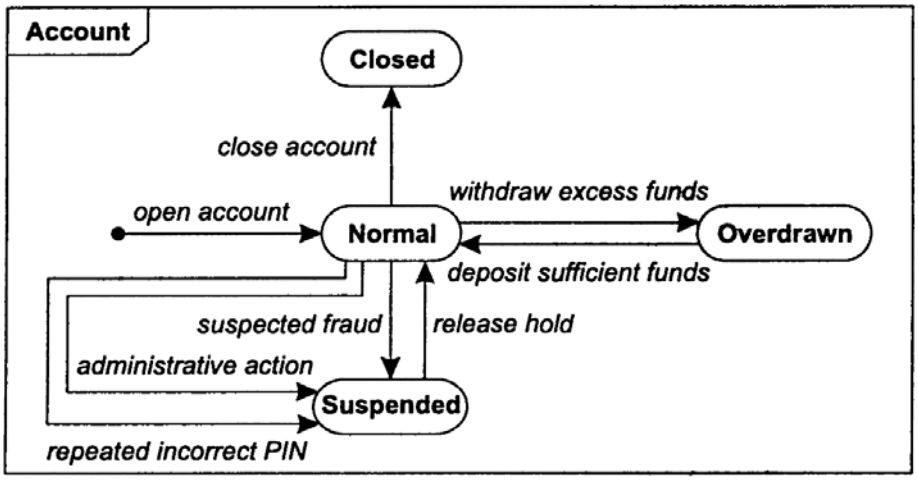


Рисунок 3.7 – Модель состояний для класса Account

Более подробные сведения о методике анализа предметной области приведены в [6, глава 12].

3.5 Анализ приложения

3.5.1 Модель взаимодействия приложения. Модель взаимодействия приложения описывает взаимодействие системы с внешним миром. Модель взаимодействия строится в несколько этапов.

Определение границы системы. Моделирование взаимодействия следует начинать с определения точной границы приложения. Вы должны решить, что будет входить в систему, а что – нет. Если граница системы очерчена корректно, во всех взаимодействиях Вы сможете рассматривать систему как черный ящик – единый объект, внутренние особенности которого скрыты от внешнего взгляда и могут быть реализованы по разному.

Идентификация действующих лиц. После определения границ системы нужно идентифицировать внешние объекты, непосредственно взаимодействующие с системой. Это и будут действующие лица. К их числу принадлежат люди, внешние устройства и другие программные системы. Самым важным качеством действующих лиц является то, что они не контролируются приложением, а их поведение должно считаться непредсказуемым. Каждое действующее лицо представляет идеализированного пользователя, который задействует какое-либо подмножество функциональности системы.

Пример с банкоматом. Для приложения банкомата действующими лицами будут Customer (Клиент), Bank (Банк) и Consortium (Консорциум).

Идентификация вариантов использования. Далее для каждого действующего лица необходимо перечислить разные способы использования системы. Каждый из этих способов называется вариантом использования (use case). Варианты использования разбивают функциональность системы на несколько дискретных единиц. Любое поведение системы должно принадлежать какому-либо варианту использования. Каждый вариант использования должен представлять

собой некий сервис (услугу), предоставляемый системой, т. е. нечто ценное для действующего лица.

На этом этапе следует нарисовать диаграмму вариантов использования. На ней следует показать действующие лица и варианты использования, соединив их между собой. Обычно вариант использования соединяют с действующим лицом, которое его инициирует, но, вообще говоря, в нем могут участвовать и другие действующие лица. Кроме того, для каждого варианта использования полезно сформулировать его краткое описание в одно-два предложения.

Пример с банкоматом. На рисунке 3.8 показаны варианты использования, краткое описание которых приведено ниже:

– initiate session (инициализация сеанса). ATM определяет личность пользователя и предлагает ему список счетов и действий;

– query account (опрос счета). Система предоставляет общие сведения о счете, такие как текущий баланс, дата последней транзакции, дата отправки последнего отчета по почте;

– process transaction (обработка транзакции). Система банкомата выполняет действие, влияющее на баланс счета, такое как вклад и перевод. Банкомат гарантирует, что все завершенные транзакции в конечном счете записываются в базу данных банка;

– transmit data (передача данных). Банкомат использует возможности консорциума для взаимодействия с компьютерами соответствующего банка.

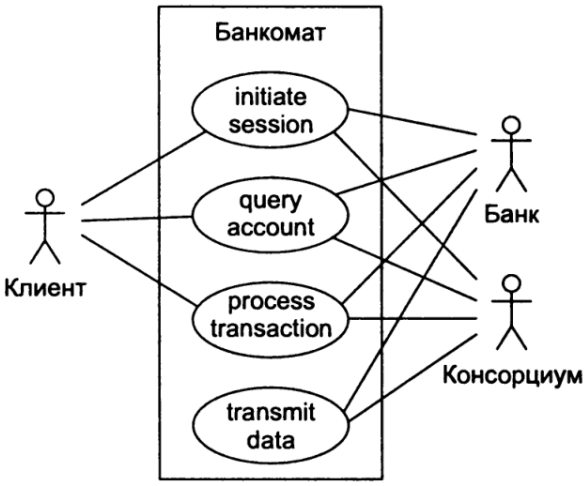


Рисунок 3.8 – Диаграмма вариантов использования для банкомата

Идентификация начальных и конечных событий. Варианты использования разбивают функциональность системы на дискретные части и показывают действующие лица, использующие каждую из этих частей. Однако поведение системы демонстрируется недостаточно четко. Для понимания поведения необходимо знать последовательности выполнения, соответствующие каждому из вариантов использования. Начинать их анализ следует с поиска событий, инициализирующих каждый из вариантов использования. Определите, какое лицо инициирует вариант использования, и определите событие, которое оно для этого

передает системе. Во многих случаях начальным событием является запрос некоторой услуги, предоставляемой системой. В других случаях начальным событием является происшествие, запускающее цепочку действий. Дайте этому событию осмысленное название, но пока не пытайтесь определить полный список его параметров.

Кроме того, следует определить конечное событие или группу событий, а также общие рамки событий, которые должны быть включены в каждый из вариантов использования. Разработчик модели должен определить границы варианта использования, установив его конечное событие.

Пример с банкоматом. Здесь мы приведем начальное и конечное событие для каждого варианта использования:

– initiate session (инициализация сеанса). Начальным событием является помещение клиентом банковской карты в банкомат. Конечных событий может быть два: либо система оставляет банковскую карту себе, либо возвращает ее обратно клиенту;

– query account (опрос счета). Начальное событие: клиент запрашивает данные о состоянии счета. Конечное событие: выдача необходимых сведений клиенту;

– process transaction (обработка транзакции). Начальное событие: клиент инициализирует транзакцию. Конечных событий может быть два: завершение или отказ транзакции;

– transmit data (передача данных). Начальным событием может быть запрос клиентом данных о состоянии счета. Кроме того, передача данных может быть инициирована после устранения неполадок в сети или перебоев с питанием. Конечное событие: успешная передача данных.

Подготовка типовых сценариев. Для каждого варианта использования нужно подготовить один или несколько типичных сценариев, чтобы почувствовать ожидаемое поведение системы. Эти сценарии описывают основные взаимодействия, форматы внешних отображаемых данных, а также обмен информацией. Сценарием называем последовательность событий на множестве взаимодействующих объектов. Анализ следует осуществлять в терминах примеров взаимодействия. Таким образом, Вы будете уверены, что общая последовательность взаимодействия получится корректной.

Для большинства задач логическая корректность зависит от взаимной последовательности взаимодействий, а не от конкретных временных промежутков между ними.

Подготовьте сценарии для типовых ситуаций – взаимодействий без необычных входных параметров и ошибочных ситуаций. Событием является обмен информацией между объектом системы и внешним агентом (пользователем или другой задачей). Параметром события является передаваемая информация. Например, параметром события «введен пароль» является сам введенный пароль. События без параметров тоже имеют значение и достаточно распространены. Само осуществление события является информацией. Для каждого события необходимо указать вызвавшее его лицо и параметры события.

Пример с банкоматом. На рисунке 3.9 приведены типовые сценарии для каждого из вариантов использования.

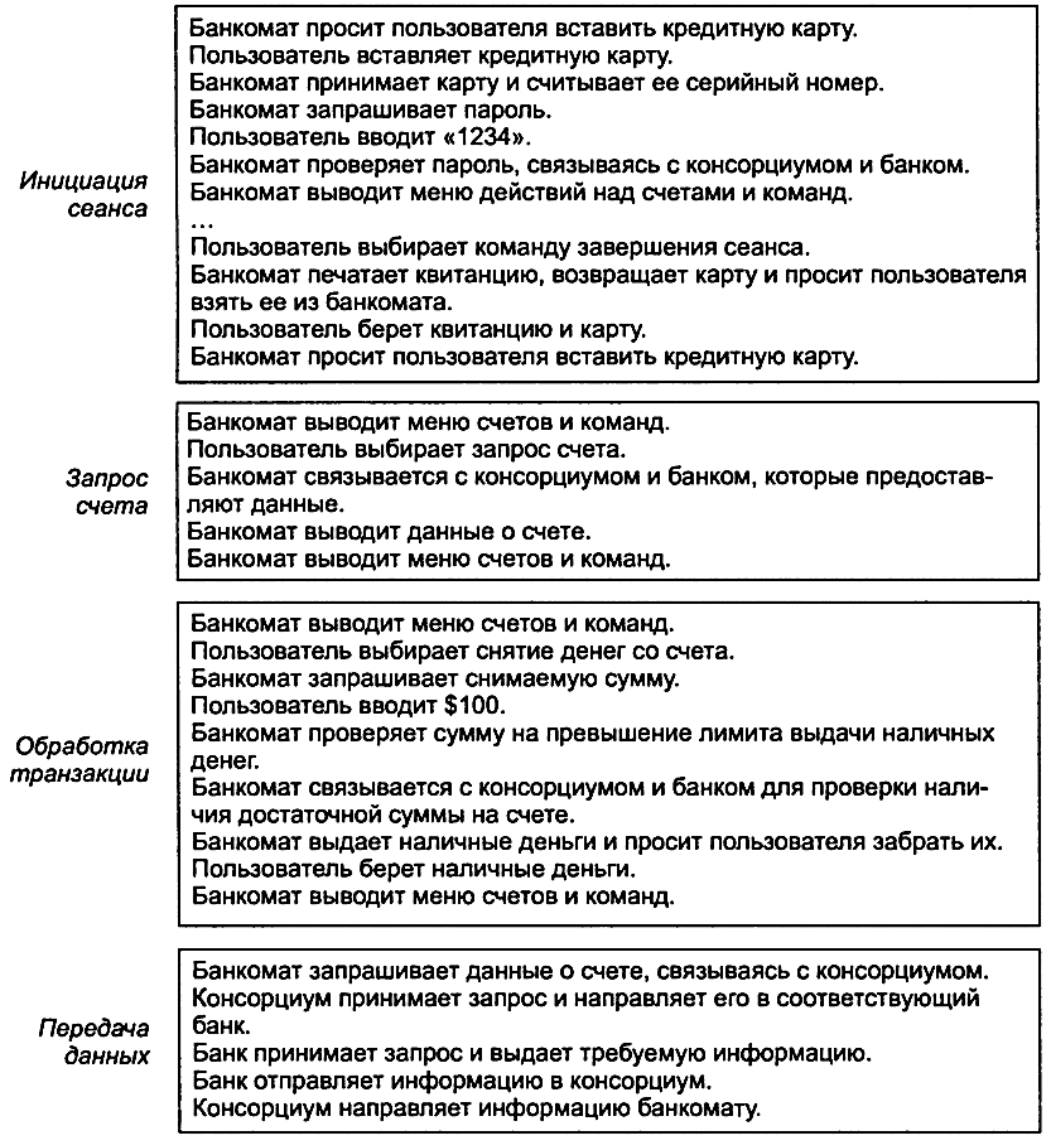


Рисунок 3.9 – Типовые сценарии для банкомата

Нетипичные сценарии и исключительные ситуации. После разработки типовых сценариев необходимо рассмотреть особые ситуации, такие как отсутствие введенных значений, ввод минимального и максимального значений, ввод одинаковых значений. Затем нужно рассмотреть ошибочные действия ситуации, включая ввод неправильных значений и отсутствие отклика. Для многих интерактивных приложений обработка ошибок является наиболее сложной частью процесса разработки. Необходимо предоставлять пользователю возможность отменить операцию или откатиться к четко определенной начальной точке каждого этапа.

Пример с банкоматом. В примере с банкоматом к нетипичным ситуациям и исключительным ситуациям можно отнести следующее:

– банкомат не может прочитать карту;

– срок действия карты истек;

– тайм-аут при ожидании банкоматом ответа клиента;

– сумма введена не верно;

– в банкомате кончились наличные или бумага;

– транзанкция отклонена из-за подозрительной схемы использования карты.

Выделение внешних событий. Проанализируйте все разработанные сценарии и выделите внешние события: ввод данных, принятие решений, прерывания и взаимодействия с другими пользователями и внешними устройствами. Передача информации объекту является событием. Например, «введен пароль» – это сообщение, переданное от внешнего агента User (Пользователь) объекту приложения ATM (Банкомат). Многие события характеризуются определенными параметрами.

Сгруппируйте под одинаковым названием события, оказывающие одинаковое влияние на поток управления, даже если значения их параметров отличаются. Например, «введен пароль» – это событие, параметром которого является значение пароля. Выбор значения пароля не влияет на поток управления, поэтому события с разными паролями являются экземплярами одного и того же типа событий. Экземпляры событий, значения которых влияют на поток управления, должны быть отнесены к разным типам событий. «Правильный счет», «неверный счет» и «неверный пароль» – разные события, которые не следует группировать под названием «состояние карты».

Подготовьте диаграмму последовательности для каждого сценария. Диаграмма последовательности показывает участников взаимодействия и порядок сообщений, которыми они обмениваются. Каждому участнику выделяется свой столбец. Диаграмма показывает отправителя и получателя каждого сообщения. Если в сценарии участвуют несколько объектов одного и того же класса, им следует присвоить разные номера. Изучив один столбец диаграммы, Вы можете определить события, непосредственно влияющие на конкретный объект. После этого Вы можете сгруппировать события, отправляемые и принимаемые каждым классом.

Пример с банкоматом. На рисунке 3.10 показана диаграмма последовательности для сценария варианта использования process transaction (обработка транзакции). На рисунке 3.11 сгруппированы события. Стрелки указывают, какой объект является отправителем, а какой получателем (для каждого сообщения). Параметры событий на рисунке 3.11 не показаны.

Подготовка диаграмм деятельности для сложных вариантов использования. Диаграммы последовательности описывают диалог и взаимодействие действующих лиц, но на них нельзя отразить имеющиеся альтернативы и принятия решения. Вам придется рисовать отдельную диаграмму для основного потока взаимодействия и отдельные диаграммы для каждой ошибочной ситуации и каждой точки принятия решения. Диаграммы деятельности позволяют объединить все это поведения благодаря документированию ветвлений и слиянию потока управления.

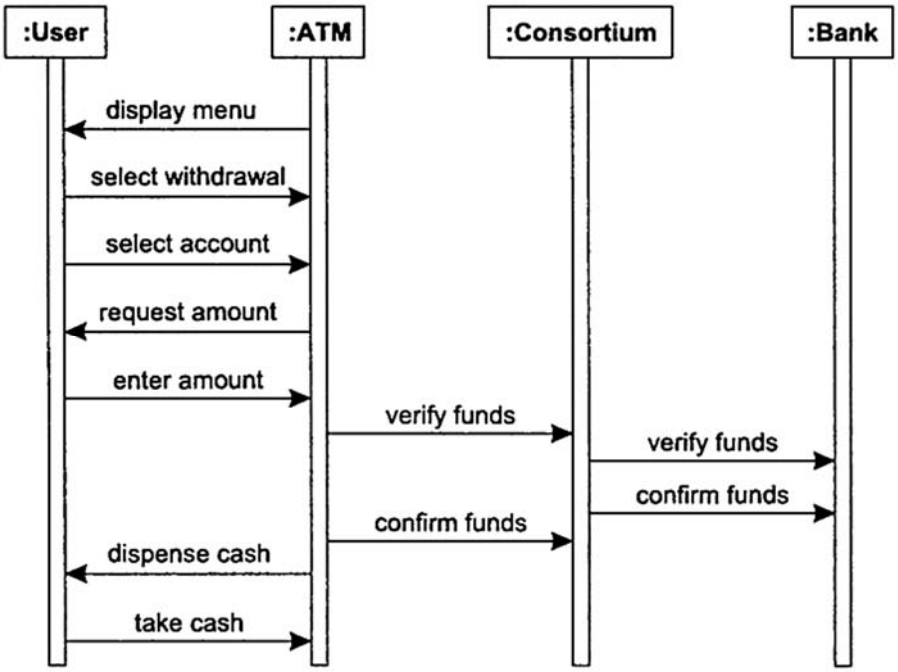


Рисунок 3.10 – Диаграмма последовательности для сценария «обработка транзакции»

Рисунок 3.11 – События для примера с банкоматом

Пример с банкоматом. На рисунке 3.12 показано, что перемещение клиентом карты в банкомат может вызвать множество различных последствий. Некоторые варианты отклика указывают на наличие проблем с картой или счетом (банкомат не возвращает карту). Только успешное прохождение проверки разрешает работу с банкоматом.

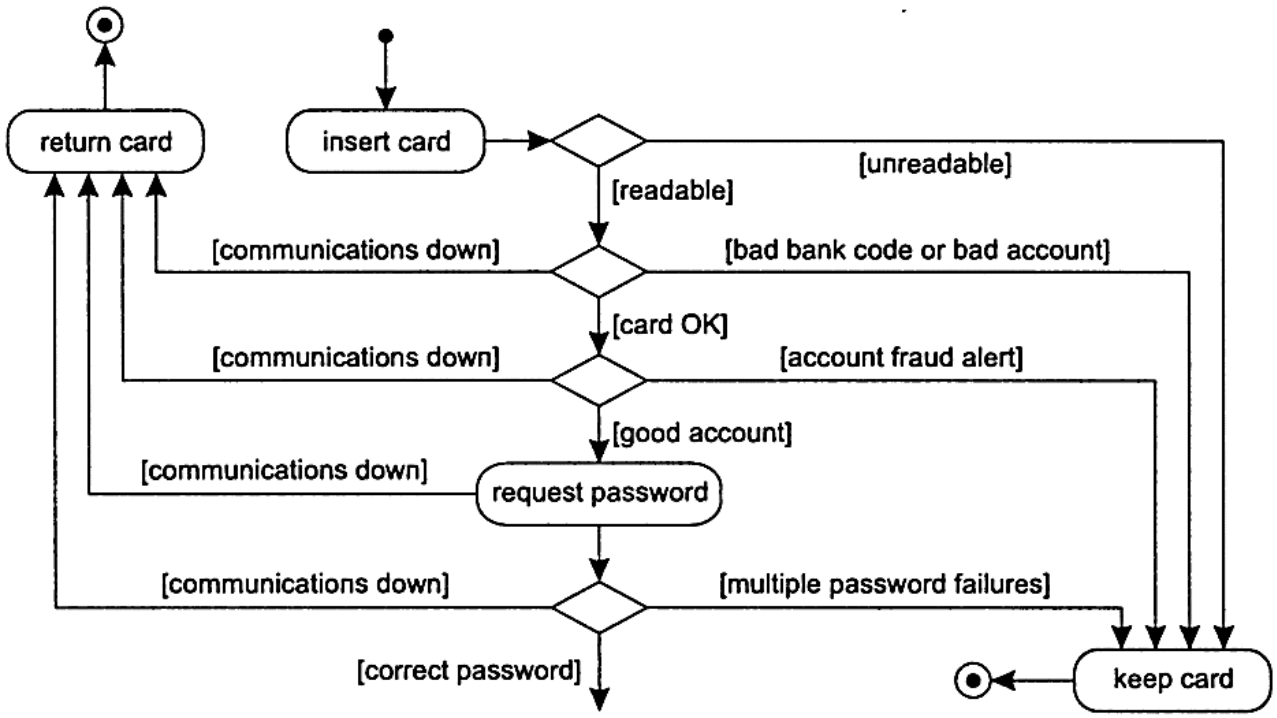


Рисунок 3.12 – Диаграмма деятельности для проверки карты

3.5.2 Модель классов приложения. Классы приложения описывают само приложение, а не объекты реального мира с которыми оно работает. Большинство классов приложения связаны с компьютерами и определяют восприятие приложения пользователями. Модель классов приложения строится в несколько этапов.

Определение интерфейсов пользователя. Пользовательским интерфейсом называется объект или группа объектов, предоставляющих пользователю системы единую точку доступа к объектам предметной области, командам и параметрам приложения. В процессе анализа внимание уделяется прежде всего потокам информации и управления, а не формату представления.

На этапе анализа пользовательский интерфейс рассматривается достаточно грубо. Не следует беспокоится о том, каким конкретно образом будет введена определенная порция данных. Вместо этого нужно попытаться определить команды, которые пользователь должен иметь возможность выполнять. Командой (command) называется крупномасштабный запрос некоторой услуги системы. Например, командами могут быть «забронировать билеты» и «найти поисковую строку в базе данных».

На этом этапе нужно набросать прототипы интерфейсов, чтобы с их помощью визуализировать работу приложения и проверить, не было ли пропущено что-нибудь важное.

Пример с банкоматом. На рисунке 3.13 показан возможный интерфейс банкомата. Точные детали на этом этапе несущественны. Самое важное – это информация, которой пользователи обмениваются с системой.

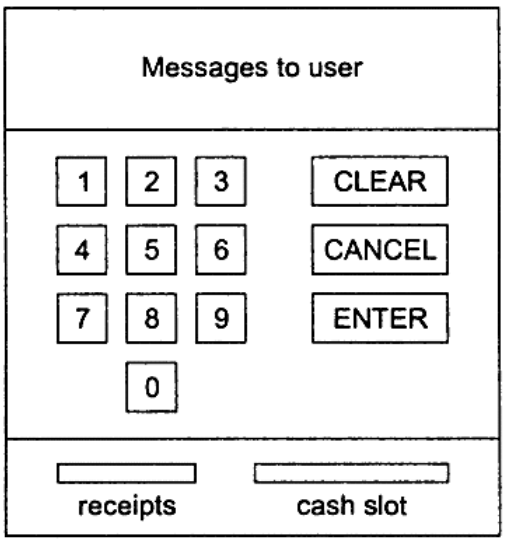


Рисунок 3.13 – Прототип интерфейса банкомата

Определение пограничных классов. Система должна быть способна работать с информацией, принимаемой из внешних источников, но ее внутренняя структура не должна зависеть от этих источников. Поэтому полезно определить пограничные классы, которые будут изолировать внутреннюю часть системы от внешнего мира. Пограничным называется класс, который является посредником во взаимодействии системы с внешним источником. Такой класс должен уметь принимать данные в формате одного или нескольких внешних источников, например, из базы данных, и преобразовывать их к формату, с которым работает внутренняя часть системы.

Пример с банкоматом. Для банкомата полезно будет определить пограничные классы CardBoundary (Граница Карты) и AccountBoundary (Граница Счета), которые скроют взаимодействие банкомата с консорциумом.

Определение управляющих объектов. Управляющий объект (controller) – это активный объект, осуществляющий управление внутри приложения. Он принимает сигналы их внешнего мира и от объектов системы, реагирует на них, вызывает операции на объектах системы и передает сигналы во внешний мир. Управляющий объект – это воплощение поведения в форме объекта. С поведением в такой форме проще работать и его проще изменять, чем простой код.

Проектирование управляющего объекта сводится, по большей части, к разработке диаграммы состояний этого объекта. В модели классов приложения существование управляющих объектов тоже необходимо отразить вместе с информацией, которую они обрабатывают, и ассоциациями, которые их связывают с другими объектами системы.

Пример с банкоматом. Из сценариев, показанных на рисунке 3.9, следует, что банкомат имеет два главных управляющих цикла. Внешний цикл проверяет клиентов и счета. Внутренний цикл обслуживает транзакции. Каждый из этих циклов наиболее естественно воплотить в управляющем объекте.

Проверка по модели взаимодействия. Построив модель классов приложения, вернитесь к вариантам использования и подумайте о том, как они могут осуществляться этим приложением. Например, если пользователь передает приложению команды, параметры команды должны передаваться каким-либо объектом пользовательского интерфейса. Запрос на саму команду должен исходить из какого-либо управляющего объекта. При наличии корректных моделей классов предметной области и приложения Вы должны быть способны имитировать варианты использования при помощи классов. Ручное моделирование помогает убедиться в том, что все составляющие находятся на своих местах.

Пример с банкоматом. На рисунке 3.14 показана предварительная модель классов приложения и классы предметной области, с которыми она взаимодействует. Интерфейсов всего два: один для пользователей, другой – для консорциума. Эти классы в модели приложения представлены заглушками, потому что пока еще не ясно, каким образом они должны быть реализованы.

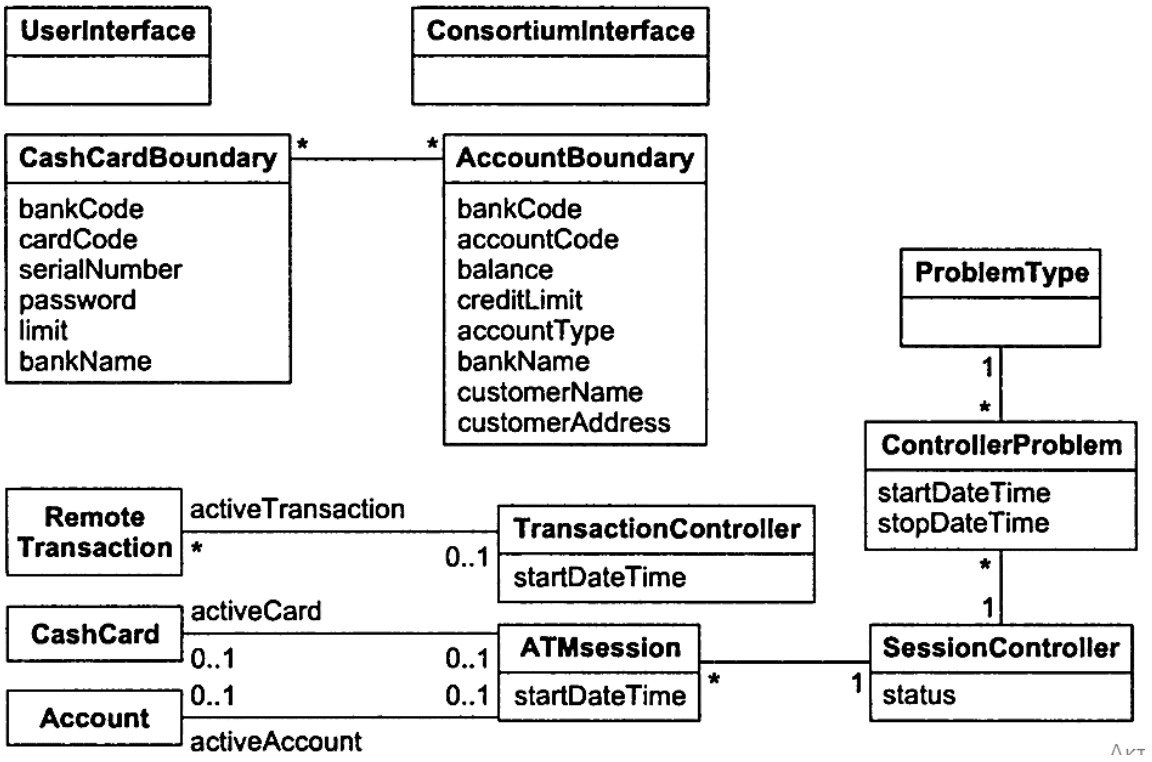


Рисунок 3.14 – Модель классов приложения

Обратите внимание, что пограничные классы объединяют информацию, полученную из разных классов предметной области. Для простоты количество пограничных классов и их отношений следует, по возможности, уменьшать.

Класс TransactionController (Управляющий Объект Транзакций) обрабатывает запросы по счетам и собственно транзакции. Класс SessionController (Управляющий Объект Сеансов) управляет классами ATMsession (Сеанс Банкомата), каждый из которых обслуживает клиента. Каждый объект ATMsession может иметь, а может и не иметь корректных CarhCard (Банковская Карта) и Account (Счет). Класс SessionController имеет атрибут status (состояние), который может

принимать значения ready (готов), impaired (ограниченный режим – кончилась бумага или наличные, но некоторые операции все равно могут быть выполнены) и down (отказ, например, линии связи). Система ведет журнал объектов ControllerProblem (Неполадка Управляющих Объектов) с указанием типов неполадок (отказ устройства считывания карт, кончились наличные и т. д.).

3.5.3 Модель состояний приложения. Модель состояний приложения описывает состояния классов приложения. Построение модели классов приложения осуществляется в несколько этапов.

Выделение классов приложения. Модель классов состоит из «компьютерных» классов, видимых пользователям и важных для работы приложения. Вы должны рассмотреть каждый из этих классов и определить, обладает ли он несколькими разными состояниями. Чаще всего состояниями обладают классы пользовательского интерфейса и управляющих объектов. Пограничные классы напротив, обычно являются статическими и осуществляют импорт и экспорт данных.

Пример с банкоматом. Классы пользовательского интерфейса не обладают состояниями. Это связано с простотой интерфейса программы. Пограничные классы также лишены состояний. Зато управляющие объекты характеризуются важными состояниями, которые мы рассмотрим ниже.

Поиск событий. Ранее мы подготовили набор сценариев для описания модели взаимодействия приложения. Теперь Вы должны изучить эти сценарии и выделить из них события.

Обратите внимание на разницу между последовательностью этапов построения моделей состояний для предметной области и для приложения. В первом случае мы начинали с поиска состояний, после чего выделяли события. Для модели приложения важно прежде всего поведение, и поэтому варианты использования раскрываются сценариями, которые содержат события.

Пример с банкоматом. Вернемся к сценариям из модели взаимодействия приложения. Там мы обнаружим события: insert card (вставка карты), enter password (ввод пароля), end session (окончание сеанса) и take card (извлечение карты).

Построение диаграмм состояний. Следующий этап состоит в построении диаграммы состояний для каждого класса приложения, обладающего существенно зависящим от времени поведением. Выберите один из классов и возьмите его диаграмму последовательности. Расставьте события, в которых участвует класс, в виде графа, над дугами которого напишите названия событий. Интервал между любыми двумя событиями будет состоянием. Дайте каждому состоянию конкретное имя, если оно получается осмысленным, а если нет – оставьте состояние безымянным. Объедините связанные между собой диаграммы последовательности в диаграмму состояний. Эта диаграмма будет последовательностью состояний и событий. Каждый сценарий или диаграмма последовательности определяют один маршрут на диаграмме состояний.

После рассмотрения всех типичных событий добавьте на диаграмму нетипичные сценарии и исключительные ситуации.

Диаграмма состояний может считаться законченной, если она покрывает все сценарии и учитывает все события, которые могут повлиять на состояния.

Пример с банкоматом. На рисунке 3.15 показана диаграмма состояний для управляющего объекта SessionController (Управляющий Объект Сеанса). Центр диаграммы описывает основную часть поведения, связанную с обработкой введенной карты и пароля. Обрыв связи может прервать обработку в любой момент. В этом случае банкомат возвращает карту. После завершения транзакции распечатка чека осуществляется одновременно с возвратом карты. Пользователь может взять карту и чек в произвольном порядке.

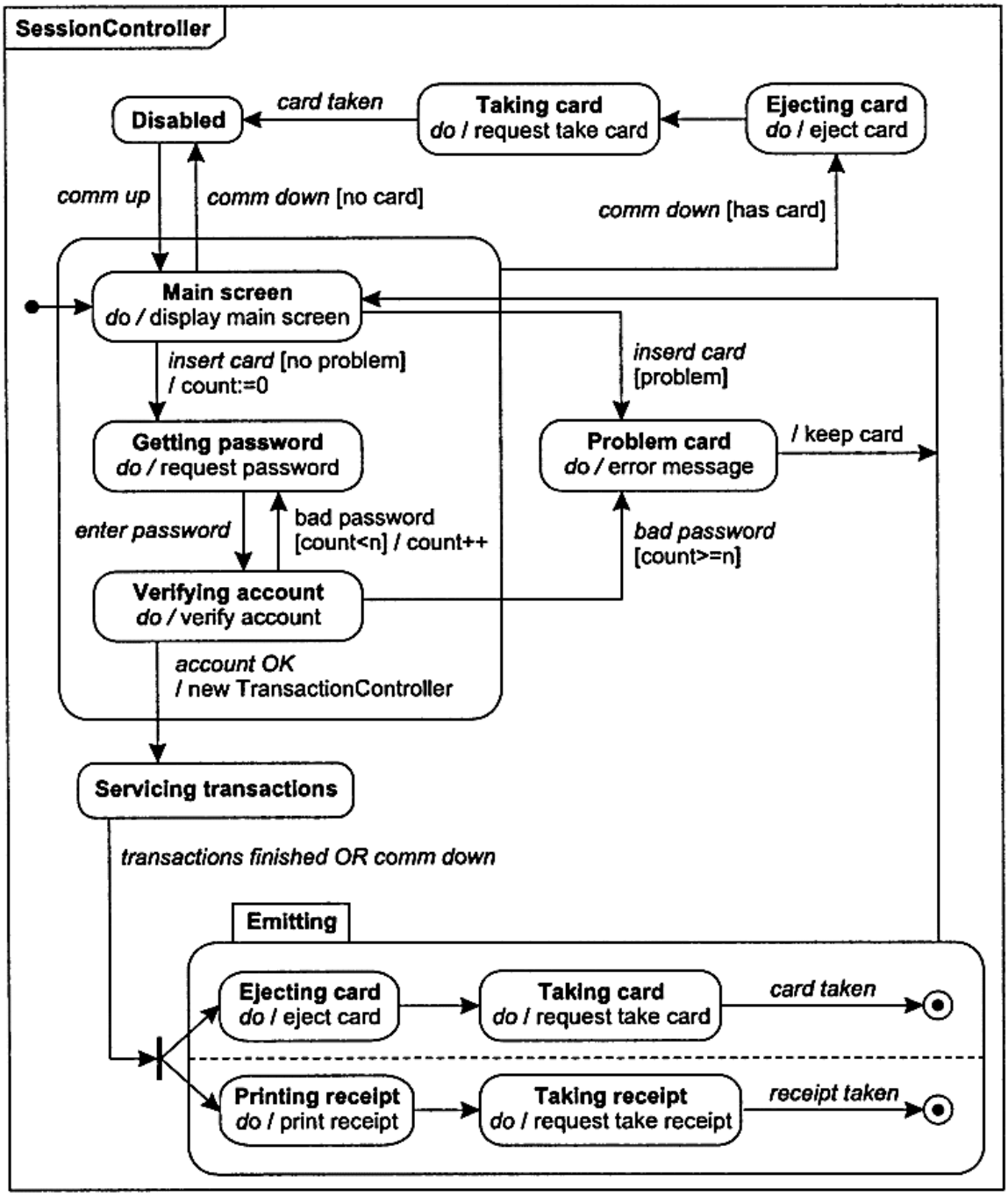


Рисунок 3.15 – Диаграмма состояний SessionController

Более подробные сведения о моделировании взаимодействия приложения содержатся в [6, глава 13].

3.6 Проектирование классов

Цель этапа проектирования классов состоит в том, чтобы довести определения классов, выделенных на этапе анализа задачи, до их их конечного вида и выбрать алгоритмы для операций. Проектирование классов делится на несколько этапов.

3.6.1 Реализация вариантов использования с помощью операций. Первый этап проектирования классов состоит в добавлении операций, реализующих варианты использования. Варианты использования определяют требуемое поведение, но не его реализацию. Проектировщик должен придумать операции, которые обеспечат поведение, заданное вариантами использования.

Начинать нужно с составления списка ответственности варианта использования или операции. Ответственностью называется нечто известное объекту или нечто такое, что он должен сделать. Ответственность не является четко определенным понятием. Это просто средство, помогающее мыслительному процессу. Например, в театральной интернет-кассе операция бронирования должна найти свободные места на выбранном спектакле, пометить их как занятые, получить платеж от покупателя, организовать доставку билетов и перевести платеж на соответствующий счет.

Каждая операция может нести ответственность за несколько действий. Некоторые виды ответственности могут быть общими у разных операций. Группируйте виды ответственности в кластеры и старайтесь делать эти кластеры согласованными. Это означает, что каждый кластер должен состоять из родственных видов ответственности, которые обслуживаются одной низкоуровневой операцией.

После этого нужно определить операцию для каждого кластера видов ответственности. Операцию следует определять так, чтобы она не ограничивалась конкретными обстоятельствами. Следует избегать и чрезмерной общности определений, потому что тогда операция получается нечеткой. Цель состоит в том, чтобы предвидеть возможное использование новой операции в будущем. Если операция может быть использована в нескольких частях текущего проекта, достаточно будет сделать ее такой общей, чтобы она покрывала все существующие варианты своего применения.

В конце назначьте низкоуровневые операции классам. Если подходящих классов найти не удается, Вам придется подумать и добавить в модель новый класс низкого уровня.

Пример с банкоматом. Один из вариантов использования назывался process transaction (обработка транзакции). Вспомните, что Transaction (Транзакция) – это множество классов Update (Обновление) и что логика операции зависит от того, является ли транзакция снятием денег со счета, помещением их на счет или переводом денег между счетами:

– снятие денег со счета (withdrwal). Операция снятия денег несет ответственность за множество действий, таких как получение суммы транзакции

от клиента, проверка наличия достаточных средств на счете, проверка соответствия указанной суммы политике банка, проверка наличия достаточных средств в банкомате, выдача наличных, уменьшение суммы на счете, печать клиентского чека. Некоторые из этих действий должны выполняться в контексте транзакции базы данных. Такая транзакция гарантирует, что либо все действия будут выполнены вместе, либо ни одно из них. Это относится, например, к выдаче наличных и уменьшению суммы на счете;

– размещение денег на счете (deposit) Операция размещения денег на счете несет ответственность за такие действия, как получение суммы транзакции от клиента, получение конверта с деньгами от клиента, печать временной метки на конверте, увеличение счета, печать клиентского чека. Некоторые действия также должны выполняться в контексте транзакции;

– трансфер (transfer). Операция несет ответственность за такие действия, как получение номера исходного счета, получение номера целевого счета, получение суммы трансфера, проверка наличия достаточных средств на исходном счете, проверка соответствия указанной суммы политике банка, уменьшение суммы на исходном счете, увеличение суммы на целевом счете, печать суммы на чеке. И здесь некоторые действия должны выполняться в контексте транзакции.

3.6.2 Разработка алгоритмов операций. На этом этапе нужно сформулировать подходящие алгоритмы каждой операции. Модели, созданные на этапе анализа задачи, описывают, что именно должна делать операция для своих клиентов, а алгоритм показывает, как это делается. Проектирование алгоритмов делится на несколько этапов.

Выбор алгоритмов. Многие операции реализуются очевидным образом, т. к. они сводятся к прослеживанию модели классов и получению или изменению значений атрибутов или связей. Для записи таких операций удобно использовать систему обозначения языка OCL [6, глава 3].

Некоторые операции не могут быть полностью выражены в виде цепочек прослеживания связей модели классов. Для описания таких ситуаций рекомендуется использовать псевдокод. Псевдокод помогает обдумывать алгоритм, не углубляясь в тонкости программирования.

Когда эффективность не слишком важна, нужно выбирать простые алгоритмы. Творческую активность лучше направить на алгоритмы тех операций, которые могут стать узкими местами приложения. Например, поиск значения в множестве из n элементов методом перебора требует в среднем $n/2$ операций, тогда как бинарный поиск выполняется за $\log n$ операций.

Пример с банкоматом. Взаимодействие между компьютером консорциума и банковскими компьютерами может быть достаточно сложным. Одна из проблем связана с распределенностью вычислений: компьютер консорциума находится в одном месте, а банковские компьютеры – во множестве других мест. Важно, чтобы компьютер консорциума допускал масштабирование: сеть банкоматов не может оправдать затраты на слишком мощный компьютер консорциума, но этот компьютер должен быть способен обслуживать новые банки по мере их присоединения к консорциуму. Третий вопрос связан с тем, что банковские

системы представляют собой отдельные приложения, существующие независимо от системы банкоматов. Это приводит к неизбежным преобразованиям и компромиссам, связанным с необходимостью поддерживать различные форматы данных. По этим причинам выбор алгоритмов, координирующих взаимодействие консорциума и банка, является достаточно важным решением.

Выбор структур данных. Алгоритмы работают с определенными структурами данных. На этапе анализа нас интересовала логическая структура информации в системе, а на этапе проектирования нужно предложить структуры данных, которые позволят реализовывать эффективные алгоритмы. Структуры данных не добавляют в модель новой информации, они служат тому, чтобы организовать эту информацию в форме, удобной для применения алгоритмов. Многие структуры данных являются экземплярами классов-контейнеров. К таким структурам относятся массив, списки, очереди, стеки, множества, словари и т. п.

Пример с банкоматом. Transaction (Транзакция) состоит из множества update (обновление). Мы должны пересмотреть модель классов – на самом деле, транзакция представляет собой последовательность обновлений, т. е. класс Transaction должен содержать упорядоченный список update.

Определение внутренних классов и операций. В процессе высокоуровневых операций Вам придется придумывать новые операции, относящиеся к более низким уровням. Некоторые низкоуровневые операции относятся к «операциям по списку». Однако чаще всего добавления новых операций избежать не удается.

Раскрытие алгоритмов может вызвать необходимость создания новых классов для хранения промежуточных результатов. Обычно эти классы отсутствуют в постановке задачи, т. к. они относятся к артефактам приложения.

Пример с банкоматом. Среди действий, относящихся к сфере ответственности варианта использования process transaction (обработка транзакции), мы выделили печать клиентского чека. Банкомат должен печатать на чеке все действия клиентов, чтобы они не забывали о них. В модели класс Receipt (Чек) отсутствовал, поэтому нам придется добавить его.

Назначение операций классам. Когда класс имеет отношение к реальному миру, его операции обычно достаточно очевидны. В процессе проектирования добавляются внутренние классы, соответствующие не объектам реального мира, а лишь некоторым его аспектам. Поскольку эти классы придумываются Вами, они являются в достаточной степени произвольными и границы между ними определяются не логической необходимостью, а удобством.

Как выбрать класс, к которому должна быть отнесена операция? Если в операции участвует только один объект, решение будет простым: нужно запросить выполнение операции у этого объекта или приказать ему выполнить ее. Если же объектов несколько, решение усложняется. Вы должны определить объект, играющий в этой операции главную роль. Задайте себе следующие вопросы:

– *получатель действия.* Есть ли такой объект, над которым выполняют операцию все остальные объекты? В большинстве случаев лучше связывать операцию с ее целью, а не с инициаторами;

– *запрос на обновление.* Есть ли такой объект, который изменяется в ходе выполнения операции, тогда как другие объекты лишь сообщают сведения о своем состоянии? Изменяющийся объект и является целью операции;

– *центральный класс.* Какой класс находится ближе всего к центру графа всех всех участвующих в операции классов и ассоциаций? Если классы и ассоциации образуют звезду вокруг одного центрального класса, этот класс является целью операции;

– *аналогия с реальным миром.* Если бы объекты существовали не в программе, а в реальном мире, какой объект вы бы толкали, перемещали или активировали, чтобы выполнить операцию?

Пример с банкоматом. Рассмотрим операции варианта использования proceess transaction (обработка транзакции) из раздела «Реализация вариантов использования с помощью операций» и попробуем назначить их различным классам:

– Customer.getAmount() – получение суммы транзакций от клиента. Значение amount будет храниться как атрибут объектов Update (Обновление), но мы предполагаем, что эти объекты недоступны в момент ввода суммы транзакции и создаются лишь после проведения ряда проверок. Мы будем хранить значение суммы транзакции во временной атрибуте;

– Account.verifyAmount(amont) – проверка наличия запрошенной суммы на балансе;

– Bank.verifyAmount(amont) – проверка соответствия суммы политике банка;

– ATM.verifyAmount(amont) – проверка наличия достаточного количества купюр в банкомате. Обратите внимание, что в нашей модели имеется несколько методов verifyAmount, принадлежащих разным классам. Это удобный способ организации модели. Все методы должны иметь одинаковую сигнатуру;

– ATM.disburseFunds(amount) – выдача наличных;

– Account.debit(amount) – уменьшение суммы на банковском счете;

– Receipt.postTransaction() – печать данных о транзакции на чеке клиента. Вам может показаться, что нам следовало связать классы Customer (Клиент) и Receipt (Чек), но модель получится более точной, если вместо этого мы свяжем классы CashCard (Банковская карта) и Receipt (Чек). При прослеживании модели одна банковская карта подразумевает одного клиента, но благодаря связи CashCard и Receipt мы можем проследить, какие экземпляры CardAuthorization (Карта авторизации) и CashCard (Банковская Карта) использовались во время конкретного сеанса;

– ATM.receiveFunds(amount) – получение конверта с деньгами от клиента. Не совсем очевидно, к какому классу следует отнести этот метод. Мы могли бы назначить его классам ATM или Customer. Мы решили назначить его классу ATM, чтобы обеспечить симметрию с операцией ATM.disburseFunds. Печать штемпеля на конверте мы включили в операцию получения конверта;

– Accont.credit(amount) – увеличение суммы на банковском счете;

– Customer.getAccount() – эта операция подразумевает получение как исходного, так и целевого счетов. Метод должен удовлетворять неявному ограни-

чеснию: одному клиенту может принадлежать несколько счетов и он обязан указывать только те счета, которые принадлежат лично ему. Обычно пользовательский интерфейс обеспечивает выполнение этого ограничения, потому что пользователю предоставляются для выбора только его собственные счета.

На рисунке 3.16 показана модель классов банкомата с добавленными в этом разделе операциями.

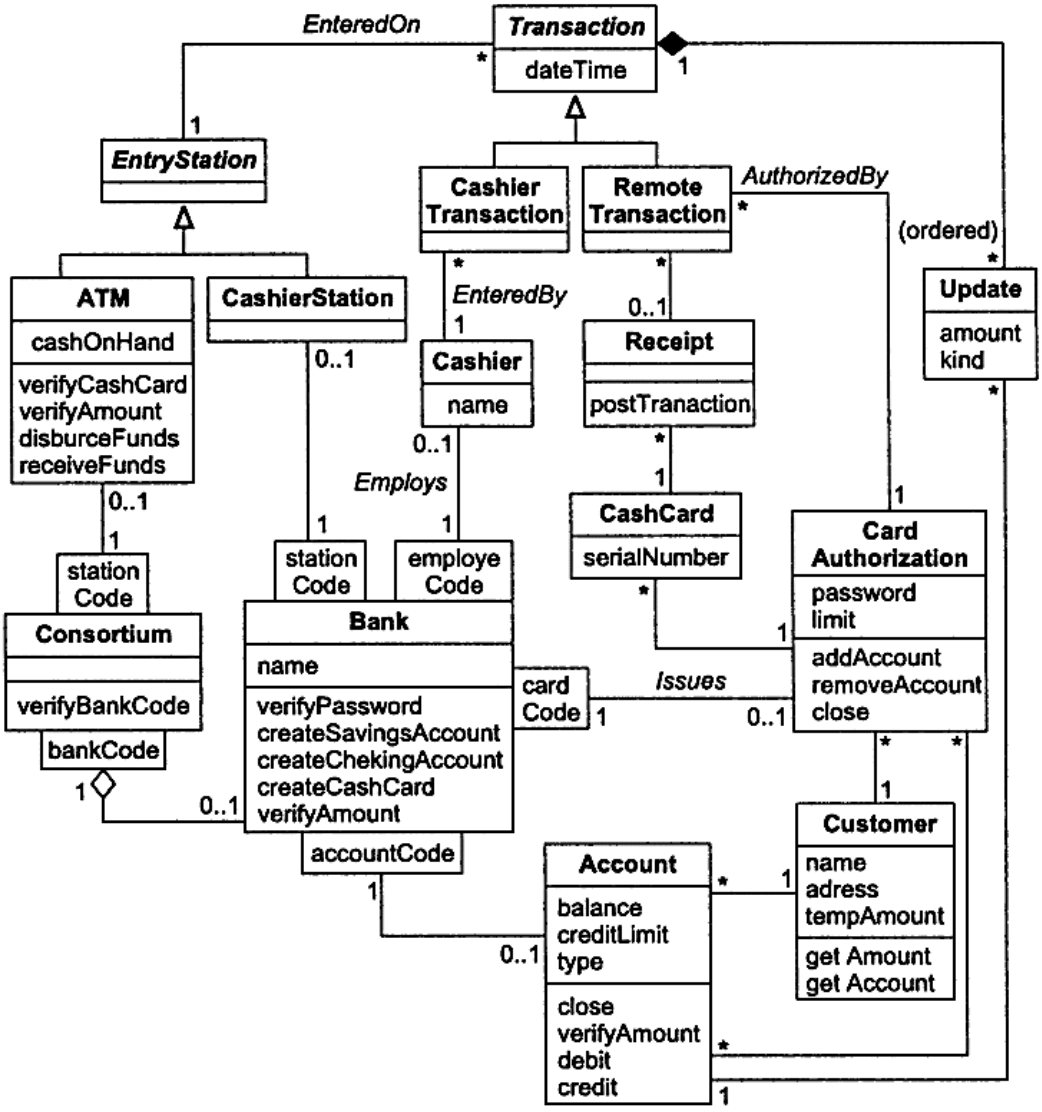


Рисунок 3.16 – Модель классов банкомата

Более подробные сведения о проектировании классов приведены в [6, глава 15].

3.7 Тестирование приложения

Тестирование состоит в выявлении и устранении ошибок в программе путем запуска ее на выполнение с такими наборами входных данных, называемых тестами, для которых результат работы программы заранее известен. Тесты подбирают так, чтобы охватить все возможные варианты работы алгоритма программы.

В разделе разрабатывается и описывается набор тестовых примеров. Набор тестов должен обеспечивать возможность проверки правильности работы программы в следующих условиях:

– проверка в нормальных условиях. Предполагает тестирование на основе данных, которые характерны для реальных условий функционирования программы;

– проверка в экстремальных условиях. Тестовые данные включают граничные значения области изменения входных переменных, которые должны восприниматься программой как правильные данные. Типичными примерами таких значений являются очень маленькие или очень большие числа и отсутствие данных. Еще один тип экстремальных условий – это граничные объёмы данных, когда массивы состоят из слишком большого числа элементов;

– проверка в исключительных ситуациях. Проводится с использованием данных, значения которых лежат за пределами допустимой области изменений, например, когда индексы массивов выходят за допустимые диапазоны, или программе, не рассчитанной на обработку отрицательных и нулевых значений переменных, в результате какой-либо ошибки приходится работать как раз с такими данными.

В записке приводятся результаты тестирования программы (таблица 3.2) и осуществляется их оценка. Оценка тестирования заключается в сравнении результатов, полученных при выполнении теста, с заранее известными контрольными значениями.

Таблица 3.2 – Шаблон таблицы для представления результатов тестирования программы

Номер теста	Входные данные	Полученный результат	Контрольное значение
1	X1, X2,X3	Y1, Y2	C1, C2, C3
2	–	–	–
3	–	–	–

Для проверки приложения на правильность функционирования рекомендуется использовать юнит-тесты. С помощью юнит-тестов должна быть проверена корректность работы только наиболее сложных методов приложения.

3.8 Описание работы приложения

Описание работы приложения подразумевает краткое словесное описание всех режимов работы приложения с использованием скриншотов окна консоли или Windows Forms.

4 Оформление пояснительной записки

Пояснительную записку следует оформлять в соответствии с требованиями ГОСТ 2.105–95 [10]. В данном разделе будут приведены основные положения этого документа.

Для оформления пояснительной записки используют писчую бумагу формата А4. Текст должен быть набран в редакторе Word и отпечатан на принтере через полуторный межстрочный интервал шрифтом Times New Roman.

Все листы пояснительной записки, включая графики, схемы, таблицы и приложения, должны содержать стандартную рамку и иметь сквозную нумерацию страниц. Титульный лист не нумеруется, а при подсчете числа страниц считается первым.

Основную надпись второй страницы оформляют по ГОСТ 2.104 (форма 2) (рисунок 4.1), а остальных страниц – по ГОСТ 2.104–68 (форма 2а) (рисунок 4.2). <Шифр> имеет следующую структуру: КП.Шифр специальности. Номер зачётки.Двузначное число, соответствующее номеру студента в списке группы. ПЗ. Здесь КП – курсовой проект; Шифр специальности – для гр. АСОИР – 090301, для гр. ПИР – 090304; ПЗ – пояснительная записка

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="4" rowspan="2"><Шифр></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Изм.</td><td>Лист</td><td>№ докум.</td><td>Подп.</td><td>Дата</td><td colspan="4"></td></tr>
<tr><td>Разраб.</td><td>Иванов</td><td></td><td></td><td rowspan="5">Разработка компьютерной игры «Морской бой»

Курсовой проект</td><td>Лит.</td><td>Лист</td><td>Листов</td></tr>
<tr><td>Пров.</td><td>Петров</td><td></td><td></td><td>У</td><td></td><td>2</td><td>24</td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="3" rowspan="3">АСОИР-201</td></tr>
<tr><td>Н.контр.</td><td></td><td></td><td></td></tr>
<tr><td>Утв.</td><td></td><td></td><td></td></tr>
</table>

Рисунок 4.1 – Основная надпись по форме 2

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td rowspan="2"><Шифр></td><td>Лист</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Изм.</td><td>Лист</td><td>№ докум.</td><td>Подп.</td><td>Дата</td><td></td><td></td></tr>
</table>

Рисунок 4.2 – Основная надпись по форме 2а

Названия заголовков разделов и подразделов следует печатать с абзацного отступа с прописной буквы без точки в конце. Переносы слов в названиях заголовков не допускаются. Наименования разделов «Введение», «Содержание» и «Список литературы» располагают симметрично по тексту. После номеров разделов и подразделов точку не ставят. Расстояние между заголовком раздела и текстом должно составлять один интервал. Расстояние между заголовками раздела и подраздела – один интервал.

Для оформления разделов «Введение», «Содержание», «Список литературы» и разделов курсового проекта использовать стиль «Заголовок 1».

Для оформления подразделов курсового проекта использовать стиль «Заголовок 2».

Для оформления основного текста пояснительной записки использовать стиль «Обычный».

Параметры оформления стиля «Заголовок1»: шрифт Times New Roman, размер шрифта 16 pt, полужирный, отступ первая 1,25 см, выравнивание по левому

краю, межстрочный интервал полуторный. Остальные неуказанные параметры форматирования должны быть равны нулю.

Параметры оформления стиля «Заголовок 2»: размер шрифта 14 пунктов, полужирный, отступ первая 1,25 см, выравнивание по левому краю, межстрочный интервал полуторный. Остальные неуказанные параметры форматирования должны быть равны нулю.

Параметры оформления стиля «Обычный»: 14 пунктов, межстрочный интервал полуторный, выравнивание по ширине, отступ первая 1,25 см, запрет висячих строк. Остальные неуказанные параметры форматирования должны быть равны нулю.

Перед каждой позицией перечисления требований, указаний, положений и т. п. ставят дефис или при необходимости ссылки в тексте записки на одно из перечислений – строчную букву, после которой ставится скобка, при этом перечисления записывают с абзаца, но с малой буквы и разделяют между собой точкой с запятой.

Иллюстрации (рисунки, графики, диаграммы) и таблицы располагают в записке непосредственно после текста, в котором они упоминаются впервые, или на следующей странице, если в указанном месте они не помещаются.

Иллюстрации следует нумеровать арабскими цифрами в пределах раздела. В этом случае номер иллюстрации состоит из номера раздела и порядкового номера иллюстрации, разделенных точкой, например «Рисунок 1.1».

Иллюстрации, при необходимости, могут иметь наименование и подрисуночный текст, поясняющий содержание рисунка. Слово «Рисунок» и наименование помещают под рисунком с абзацного отступа. Точка в конце подписи к рисунку не ставится. Пример оформления рисунка приведен на рисунке 4.3.

Рисунок 4.3 – Пример оформления рисунка

Таблицы следует нумеровать арабскими цифрами в пределах раздела. В этом случае номер таблицы состоит из номера раздела и порядкового номера таблицы, разделенных точкой, например «Таблица 1.1». Таблицы, при необходимости, могут иметь название, которое располагают через дефис за номером таблицы, например «Таблица 1.1 – Исходные данные». Точка после наименования таблицы не ставится. Примеры оформления таблиц – см. раздел 3, таблицы 3.1 и 3.2.

Параметры оформления стиля подписи рисунков и названия таблиц должны быть следующими: шрифт Times New Roman, 12 pt, выравнивание по левому краю, межстрочный интервал полуторный, отступ первая 1,25 см.

На все иллюстрации и таблицы в тексте должны быть даны ссылки, при этом слова «Рисунок» и «Таблица» пишутся полностью, например «из рисунка 2.1 следует…», «в таблице 1.2 приведены…».

При использовании формул, научно-технических положений, стандартов и других данных необходимо делать ссылку на литературный источник, указывая его номер из списка литературы в квадратных скобках. Список источников составляется либо по алфавиту, либо по мере появления ссылок в тексте пояснительной записки.

Формулы следует записывать в общем виде с новой строки и абзацного отступа. Если формула одна и требуется пояснение символов, входящих в формулу, то за формулой ставят запятую, а если пояснений символов формулы не требуется, то ставят точку. Формулы, следующие одна за другой и не разделенные текстом, разделяют запятой. При этом за последней формулой ставят либо запятую, если необходима расшифровка символов формул, либо точку, если символы формул не надо расшифровывать.

Все формулы должны нумероваться в пределах раздела арабскими цифрами. В этом случае номер формулы состоит из номера раздела и порядкового номера формулы, разделенной точкой, например, в формуле (3.1).

Формулы рекомендуется оформлять с помощью встроенного в текстовый редактор MS Word средства, которое можно вызвать на вкладке «Вставка – Символы – Формула», либо с помощью редактора формул MathType. Рекомендуется оставить для формул установки по умолчанию.

Пояснения символов, входящих в формулы, если они не пояснены ранее в тексте, должны быть приведены непосредственно под формулой. Пояснения каждого символа следует давать с новой строки в той последовательности, в которой символы приведены в формуле. Первая строка пояснения должна начинаться словом «где», после которого ставят пробел, затем приводят обозначение символа и через дефис дают описание физического смысла символа с указанием его размерности.

Пример оформления формулы. Критерий завершения итерационного процесса вычисляется по формуле

$$|x_k - x_{k+1}| \le \varepsilon, \qquad (4.1)$$

где x_k – значение аргумента функции на k-м шаге;

x_{k+1} – значение аргумента функции на $k+1$-м шаге;

ε – погрешность решения уравнения.

Приложения оформляют как продолжение курсового проекта. Каждое приложение должно начинаться с новой страницы с указанием вверху посередине

страницы слова «Приложение» и его обозначения, а под ними в круглых скобках строчными буквами указывают вид приложения (обязательное, рекомендуемое или справочное). Далее с новой строки симметрично относительно текста записывают с прописной буквы заголовок приложения. Приложения обозначают строчными буквами русского алфавита, начиная с буквы А. При ссылках на приложения в тексте следует писать, например, «в соответствии с приложением А….».

Список литературных источников, использованных при выполнении курсового проекта, составляется либо по алфавиту, либо по мере появления ссылок в тексте пояснительной записки, и оформляется в соответствии с требованиями ГОСТ 7.1–2003 *Библиографическая запись. Библиографическое описание. Общие требования и правила оформления*. Примеры оформления списка использованных источников приведены в приложении Г.

5 Защита курсового проекта

К защите допускаются курсовые проекты, выполненные в установленные сроки и имеющие положительный отзыв руководителя о ходе проектирования.

Защита (публичная защита) курсовых проектов проводится на открытых заседаниях приемной комиссии, состоящей из двух преподавателей кафедры в сроки, регламентируемые учебным планом специальности и установленные кафедрой.

Защита включает в себя доклад студента и демонстрацию разработанного программного продукта.

В выступлении следует сформулировать цели и задачи курсовой работы, раскрыть его структуру, показать используемые при проектировании решения. Следует уделить внимание выводам, предложениям, рекомендациям, сделанным автором на основе проведенной работы. Длительность выступления составляет 5…7 мин.

После доклада члены комиссии и присутствующие на защите лица задают студенту вопросы, связанные с проектированием. Далее студент демонстрирует работу разработанного программного продукта.

По окончании защиты члены комиссии на закрытом заседании коллективно обсуждают итоги защиты каждого проекта и оценивают ее большинством голосов по стобалльной шкале в соответствии с критериями оценки курсовых проектов, изложенными в разделе 7. При равном количестве голосов приоритетное право решения предоставляется руководителю проекта.

Итоги обсуждения объявляются открыто.

В тех случаях, когда защита курсовой работы признается комиссией неудовлетворительной, студент может представить к повторной защите тот же проект с доработкой, определяемой комиссией, или же обязан разработать новую тему, которая устанавливается кафедрой. Сроки и условия защиты проекта в этом случае устанавливаются кафедрой по согласованию с деканом факультета.

6 Обязательные требования к курсовому проекту

1 Использование CASE-средств для проектирования UML-диаграмм иерархии классов, диаграмм использования (UseCase), диаграмм взаимодействия, таких, как Enterprise Architect, Rational Rose, IntelliJIDEa, NetBeans и т. п.

2 Использование полностью объектно-ориентированных сред разработки объектно-ориентированных систем. Такими средами являются, например, Microsoft Visual Studio C#, IntelliJIDEa, NetBeans, Eclipse, Delphi и др.

3 Программный продукт будет удовлетворять принципам объектно-ориентированной парадигмы программирования при выполнении следующих условий:

– базовыми элементами программного продукта являются объекты, а не алгоритмы:

– каждый объект является экземпляром определенного класса;

– классы используют инкапсуляцию;

– классы образуют иерархическую структуру;

– классы создают условия возникновения полиморфизма;

– использование принципов SOLID при создании классов.

7 Критерии оценки курсового проекта

Итоговая оценка курсового проекта представляет собой сумму баллов за его выполнение и защиту и выставляется в соответствии со шкалой (рисунок 7.1).

Оценка	Отлично	Хорошо	Удовлетворительно	Неудовлетворительно
Баллы	87–100	65–86	51–64	0–50

Рисунок 7.1 – Шкала для оценивания курсового проекта

Критериями, влияющими на результирующую оценку, являются:

1) полнота реализации требований к программе;

2) тщательность тестирования программных модулей;

3) удобство программного интерфейса;

4) стиль написания программного кода;

5) качество оформления пояснительной записки;

6) полнота и правильность ответов на вопросы;

7) соблюдение календарного плана выполнения работы.

Оценка «отлично» выставляется за работу, выполненную в установленные сроки, полностью отвечающую требованиям (раздел 6) и критериям оценки 1–7 из текущего раздела.

Оценка «хорошо» выставляется за работу, выполненную в установленные сроки, в случае, если не выполняется максимум одно или два из требований 1 или 2 (см. раздел 6), или качество программного продукта не удовлетворяет критериям оценки 1–6. Эта оценка уточняется в зависимости от качества работы в соответствии с критериями 1–6.

Оценка «удовлетворительно» выставляется в случае, если программный продукт не удовлетворяет требованиям 3 (см. раздел 6) и критерию 1 или если по сумме всех требований и критериев выполненная работа не может быть оценена как хорошая. Эта оценка уточняется в зависимости от качества работы в соответствии с критериями 1–6.

Оценка «неудовлетворительно» выставляется в случае, если сроки выполнения этапов не соблюдены, работа выполнена частично, объектно-ориентированная парадигма в части требований 3 (см. раздел 6) реализована неправильно, неточно или не реализована.

Список литературы

1 Объектно-ориентированное программирование: методические указания к выполнению курсовой работы «Программирование» / Сост. В. В. Понамарев. – Озерск: ОТИ НИЯУ МИФИ, 2014. – 28 с.

2 Методические указания по выполнению курсовой работы по дисциплине «Технология разработки программных систем» / Сост. Г. С. Иванова и [др.]. – Москва: МГТУ им. Н. Э. Баумана, 2018. – 38 с.

3 Программирование: методические рекомендации к курсовому проектированию для студентов специальности 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» / Сост. К. В. Овсянников. – Могилёв, Белорус.-Рос. ун-т, 2017. – 28 с.

4 Методические указания к курсовой работе по дисциплине «Объектно-ориентированное программирование» / Сост. А. Ю. Лексин. – Владимир: ВГУ, 2012. – 20 с.

5 **Маклафлин, Б.** Объектно-ориентированный анализ и проектирование / Б. Маклафлин, Г. Поллайс, Д. Уэст. – Санкт-Петербург: Питер, 2013. – 608 с.

6 **Рамбо, Дж.** Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – Санкт-Петербург: Питер, 2007. – 544 с.

7 **Лафоре, Р.** Объектно-ориентированное программирование в C++ / Р. Лафоре. – Санкт-Петербург: Питер, 2013. – 923 с.

8 **Дейтел, Х. М.** Как программировать на C++ / Х. М. Дейтел, П. Дж. Дейтел. – Москва: Бином-Пресс, 2008. – 1456 с.

9 **Воронина, В. В.** Технологии автоматизации бизнес-процессов предприятий / В. В. Воронина. – Ульяновск: УлГТУ, 2013. – 204 с.

10 **ГОСТ 2.105–95.** Общие требования к текстовым документам. – Москва: Изд-во стандартов, 1995. – 37 с.: ил.

Приложение А
(справочное)

Образец оформления титульного листа

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра
«Программное обеспечение информационных технологий»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту на тему

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ
«АВТОМАТИЧЕСКАЯ ТЕЛЕФОННАЯ СТАНЦИЯ»

Исполнитель студент гр. АСОИР–201
_____
Руководитель
_____

Могилев 2022

Приложение Б
(справочное)

Постановка задачи на разработку информационной системы «Сеть банкоматов»

Требуется разработать программное обеспечение, позволяющее клиентам получать доступ к банковской компьютерной системе и выполнять транзанкции без участия банковских служащих (рисунок Б.1) [6, раздел 11.3].

Банкоматы могут совместно использоваться группой банков. Каждый банк предоставляет свой компьютер для учета своих счетов и обработки транзанкций с ними. Кассиры также принадлежат отдельным банкам и взаимодействуют непосредственно с банковскими компьютерами. Кассиры вводят номера счетов и данные транзанкций вручную.

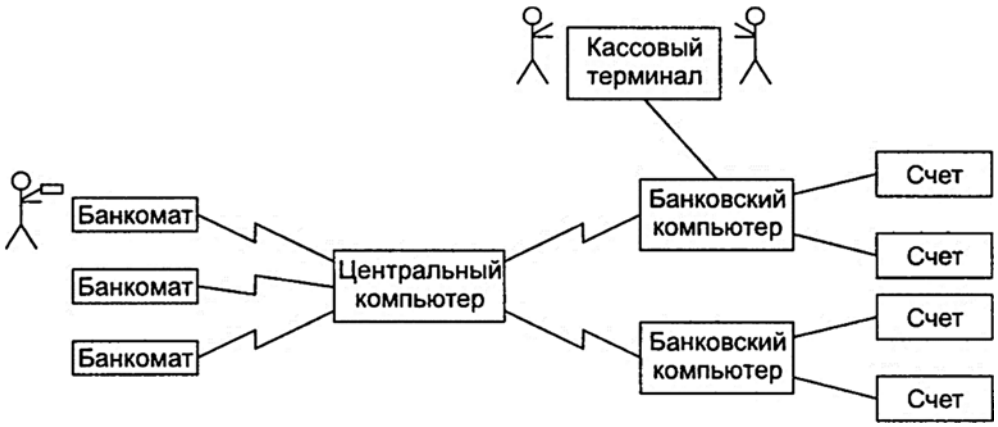


Рисунок Б.1 – Сеть банкоматов

Банкоматы взаимодействуют с центральным компьютером, который осуществляет транзанкции с соотвествующими банками. Банкомат принимает от клиента его карту, взаимодействет с клиентом, соединяется с центральной системой для выполнения транзанкции, выдает наличные и печатает чеки. Система требует ведения записей и обеспечения безопасности. Система должна корректно обрабатывать одновременное обращение к одному и тому же счету.

Приложение В
(справочное)

Постановка задачи на разработку информационной системы «Успеваемость студентов»

Требуется разработать приложение, позволяющее обрабатывать данные о студентах и их успеваемости, представленные в определенном формате. Приложение должно содержать средства для ввода и редактирования данных, обеспечивать сортировку по одному или нескольким полям данных, обеспечивать возможность поиска по одному или нескольким критериям, выполнять обработку данных и выдавать результаты обработки.

Информация о студентах должна быть представлена следующими данными:

– фамилия, имя и отчество;

– дата рождения;

– пол;

– специальность;

– курс;

– группа;

– количество экзаменов;

– оценки, полученные на экзамене.

Для хранения и обработки этой информации создать базу данных в виде набора файлов прямого доступа.

Необходимо предусмотреть возможность упорядочения данных (сортировку) по следующим полям:

– по ФИО;

– по курсу;

– по группе.

Необходимо также иметь возможность просмотра данных в виде, отсортированном по любой совокупности перечисленных полей.

Критериями поиска для данной информационной системы являются ФИО, курс, группа.

Приложение должно обеспечивать возможность решения следующих задач:

– подготовка к печати списка студентов учебной группы по запросу для одной группы и для всех групп;

– вычисление среднего балла для каждого студента; подготовка к выводу результатов в виде, отсортированном по курсу и группе, а в пределах группы предусмотреть одну из возможных сортировок: в алфавитном порядке фамилий или по убыванию среднего балла;

– для каждой учебной группы вычисление количества студентов и среднего балла, предусмотреть сортировку по курсу и номеру группы.

Для каждой из перечисленных задач предусмотреть возможность просмотра результатов на экране и возможность сохранения их в текстовых файлах для последующего использования.

Приложение Г
(справочное)

Примеры библиографических описаний

Книга. Однотомное издание

Одного автора

Тайц Б. А. Точность и контроль зубчатых колес. – М.: Машиностроение, 1972. – 368 с.: ил.

Двух и трех авторов

Дмитриевский Г. В. Эксплуатация устройств дистанционного управления разъединителями / Г. В. Дмитриевский, В. В. Курганов, М. А. Турлянский. – М.: Транспорт, 1974. – 64 с.: ил.

Четырех авторов

Теория автоматизированного электропривода / М. Г. Чиликин, В. И. Ключев, А. С. Сандлер, И. П. Копылов. – М.: Энергия, 1979. – 575 с.: ил.

Пяти и более авторов

Проектирование трансмиссий автомобилей / А. И. Гришкевич, Б. У. Бусел, Г. Ф. Бутусов и др.; под общ. ред. А. И. Гришкевича. – М.: Машиностроение, 1984. – 250 с.: ил.

Книга. Многотомное издание

Издание в целом

Савельев И. В. Курс общей физики: учеб. пособие для студентов втузов. – 2-е изд., перераб. – М.: Наука, 1982. – Т. 1–3.

Отдельный том

Станочные приспособления: справочник в 2 т. / Под ред. Б. Н. Вардашкина, А. А. Шатилова. – М.: Машиностроение, 1984. – Т. 1. – 592 с.: ил.

Учебное (справочное) пособие, справочник

Технологическая оснастка. Проектирование поводковых устройств: учеб. пособие / М. Ф. Пашкевич, Г. Я. Беляев, Ж. А. Мрочек и др.; под ред. М. Ф. Пашкевича. – Мн.: БГПА, 1992. – 183 с.: ил.

Яуре А. Г. Крановый электропривод: справочник / А. Г. Яуре, Е. М. Певзнер. – М.: Энергоатомиздат, 1988. – 253 с.: ил.

Кузьмин А. В. Расчеты деталей машин: справ. пособие / А. В. Кузьмин, И. М. Чернин, Б. С. Козинцов. – Мн.: Выш. шк., 1986. – 402 с.: ил.