# Министерство образования Республики Беларусь

УПРАВЛЕНИЕ ОБРАЗОВАНИЯ МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«Могилевский государственный политехнический колледж»

# Домашняя контрольная работа №1

По дисциплине: «**Технология разработки программного обеспечения**»

Группа ПО-455

## Выполнил О. Е. Листопадова

Шифр 21

# 202129

# 29 Что называют шаблонами (паттернами)?

Паттерн проектирования — это часто встречающееся решение определённой проблемы при проектировании архитектуры программ.

В отличие от готовых функций или библиотек, паттерн нельзя просто взять и скопировать в программу. Паттерн представляет собой не какой-то конкретный код, а общую концепцию решения той или иной проблемы, которую нужно будет ещё подстроить под нужды вашей программы.

Паттерны часто путают с алгоритмами, ведь оба понятия описывают типовые решения каких-то известных проблем. Но если алгоритм — это чёткий набор действий, то паттерн — это высокоуровневое описание решения, реализация которого может отличаться в двух разных программах.

Если привести аналогии, то алгоритм — это кулинарный рецепт с чёткими шагами, а паттерн — инженерный чертёж, на котором нарисовано решение, но не конкретные шаги его реализации.

Из чего состоит паттерн?

Описания паттернов обычно очень формальны и чаще всего состоят из таких пунктов:

проблема, которую решает паттерн;

мотивации к решению проблемы способом, который предлагает паттерн;

структуры классов, составляющих решение;

примера на одном из языков программирования;

особенностей реализации в различных контекстах;

связей с другими паттернами.

Такой формализм в описании позволил создать обширный каталог паттернов, проверив каждый из них на состоятельность.

Паттерны отличаются по уровню сложности, детализации и охвата проектируемой системы. Проводя аналогию со строительством, вы можете повысить безопасность перекрёстка, поставив светофор, а можете заменить перекрёсток целой автомобильной развязкой с подземными переходами.

Концепцию паттернов впервые описал Кристофер Александер в книге «Язык шаблонов. Города. Здания. Строительство». В книге описан «язык» для проектирования окружающей среды, единицы которого — шаблоны (или паттерны, что ближе к оригинальному термину patterns) — отвечают на архитектурные вопросы: какой высоты сделать окна, сколько этажей должно быть в здании, какую площадь в микрорайоне отвести под деревья и газоны.

Идея показалась заманчивой четвёрке авторов: Эриху Гамме, Ричарду Хелму, Ральфу Джонсону, Джону Влиссидесу. В 1994 году они написали книгу «Приемы объектно-ориентированного проектирования. Паттерны проектирования», в которую вошли 23 паттерна, решающие различные проблемы объектно-ориентированного дизайна. Название книги было слишком длинным, чтобы кто-то смог всерьёз его запомнить. Поэтому вскоре все стали называть её «book by the gang of four», то есть «книга от банды четырёх», а затем и вовсе «GoF book».

С тех пор были найдены десятки других объектных паттернов. «Паттерновый» подход стал популярен и в других областях программирования, поэтому сейчас можно встретить всевозможные паттерны и за пределами объектного проектирования.

Нужда в паттернах появляется тогда, когда люди выбирают для своего проекта язык программирования с недостаточным уровнем абстракции. В этом случае, паттерны — это костыль, который придаёт этому языку суперспособности.

Паттерн представляет определенный способ построения программного кода для решения часто встречающихся проблем проектирования. В данном случае предполагается, что есть некоторый набор общих формализованных проблем, которые довольно часто встречаются, и паттерны предоставляют ряд принципов для решения этих проблем.

Хотя идея паттернов как способ описания решения распространенных проблем в области проектирования появилась довольно давно, но их популярность стала расти во многом благодаря известной работе четырех авторов Эриха Гаммы, Ричарда Хелма, Ральфа Джонсона, Джона Влиссидеса, которая называлась "Design Patterns: Elements of Reusable Object-Oriented Software" (на русском языке известна как "Приемы объектно-ориентированного проектирования. Паттерны проектирования") и которая вышла в свет в 1994 году. А сам коллектив авторов нередко называют "Банда четырёх" или Gang of Four или сокращенно GoF. Данная книга по сути являлась первой масштабной попыткой описать распространенные способы проектирования программ. И со временем применение паттернов стало считаться хорошей практикой программирования.

Что же дает нам применение паттернов? При написании программ мы можем формализовать проблему в виде классов и объектов и связей между ними. И применить один из существующих паттернов для ее решения. В итоге нам не надо ничего придумывать. У нас уже есть готовый шаблон, и нам только надо его применить в конкретной программе.

Причем паттерны, как правило, не зависят от языка программирования. Их принципы применения будут аналогичны и в C#, и в Jave, и в других языках. Хотя в рамках данного руководства мы будем говорить о паттернах в контексте языка C#.

Также мышление паттернами упрощает групповую разработку программ. Зная применяемый паттерн проектирования и его основные принципы другому программисту будет проще понять его реализацию и использовать ее.

В то же время не стоит применять паттерны ради самих паттернов. Хорошая программа предполагает использование паттернов. Однако не всегда паттерны упрощают и улучшают программу. Неоправданное их использование может привести к усложнению программного кода, уменьшению его качества. Паттерн должен быть оправданным и эффективным способом решения проблемы.

Существует множество различных паттернов, которые решают разные проблемы и выполняют различные задачи. Но по своему действию их можно объединить в ряд групп. Рассмотрим некоторые группы паттернов. В основу классификации основных паттернов положена цель или задачи, которые определенный паттерн выполняет.

### Порождающие паттерны

Порождающие паттерны — это паттерны, которые абстрагируют процесс инстанцирования или, иными словами, процесс порождения классов и объектов. Среди них выделяются следующие:

* Абстрактная фабрика (Abstract Factory)
* Строитель (Builder)
* Фабричный метод (Factory Method)
* Прототип (Prototype)
* Одиночка (Singleton)

Другая группа паттернов - структурные паттерны - рассматривает, как классы и объекты образуют более крупные структуры - более сложные по характеру классы и объекты. К таким шаблонам относятся:

* Адаптер (Adapter)
* Мост (Bridge)
* Компоновщик (Composite)
* Декоратор (Decorator)
* Фасад (Facade)
* Приспособленец (Flyweight)
* Заместитель (Proxy)

Третья группа паттернов называются поведенческими - они определяют алгоритмы и взаимодействие между классами и объектами, то есть их поведение. Среди подобных шаблонов можно выделить следующие:

* Цепочка обязанностей (Chain of responsibility)
* Команда (Command)
* Интерпретатор (Interpreter)
* Итератор (Iterator)
* Посредник (Mediator)
* Хранитель (Memento)
* Наблюдатель (Observer)
* Состояние (State)
* Стратегия (Strategy)
* Шаблонный метод (Template method)
* Посетитель (Visitor)

Существуют и другие классификации паттернов в зависимости от того, относится паттерн к классам или объектам.

Паттерны классов описывают отношения между классами посредством наследования. Отношения между классами определяются на стадии компиляции. К таким паттернам относятся:

* Фабричный метод (Factory Method)
* Интерпретатор (Interpreter)
* Шаблонный метод (Template Method)
* Адаптер (Adapter)

Другая часть паттернов - паттерны объектов описывают отношения между объектами. Эти отношения возникают на этапе выполнения, поэтому обладают большей гибкостью. К паттернам объектов относят следующие:

* Абстрактная фабрика (Abstract Factory)
* Строитель (Builder)
* Прототип (Prototype)
* Одиночка (Singleton)
* Мост (Bridge)
* Компоновщик (Composite)
* Декоратор (Decorator)
* Фасад (Facade)
* Приспособленец (Flyweight)
* Заместитель (Proxy)
* Цепочка обязанностей (Chain of responsibility)
* Команда (Command)
* Итератор (Iterator)
* Посредник (Mediator)
* Хранитель (Memento)
* Наблюдатель (Observer)
* Состояние (State)
* Стратегия (Strategy)
* Посетитель (Visitor)

И это только некоторые основные паттерны. А вообще различных шаблонов проектирования гораздо больше. Одни из них только начинают применяться, другие являются популярными на текущий момент, а некоторые уже менее распространены, чем раньше.

### Как выбрать нужный паттерн?

Прежде всего при решении какой-нибудь проблемы надо выделить все используемые сущности и связи между ними и абстрагировать их от конкретной ситуации. Затем надо посмотреть, вписывается ли абстрактная форма решения задачи в определенный паттерн. Например, суть решаемой задачи может состоять в создании новых объектов. В этом случае, возможно, стоит посмотреть на порождающие паттерны. Причем лучше не сразу взять какой-то определенный паттерн - первый, который показался нужным, а посмотреть на несколько родственных паттернов из одной группы, которые решают одну и ту же задачу.

При этом важно понимать смысл и назначение паттерна, явно представлять его абстрактную организацию и его возможные конкретные реализации. Один паттерн может иметь различные реализации, и чем чаще вы будете сталкиваться с этими реализациями, тем лучше вы будете понимать смысл паттерна. Но не стоит использовать паттерн, если вы его не понимаете, даже если он на первый взгляд поможет вам в решении задачи.

И в конечном счете надо придерживаться принципа KISS (Keep It Simple, Stupid) - сохранять код программы по возможности простым и ясным. Ведь смысл паттернов не в усложнении кода программы, а наоборот в его упрощении.

# 38 ВУЗ

# Создайте функциональную модель деятельности ВУЗа, учитывая работу ВУЗа как по основным направлениям деятельности: обеспечение учебного процесса, научной работы, так и по дополнительным про-цессам: международная деятельность, работа по договорам, социальная работа.

Контекстная диаграмма отражает отношение системы с внешней средой. Она содержит только один процесс, обозначающий в целом деятельность всей моделируемой системы. В данном случае процесс будет называться «Работа фирмы».

Процесс (работа) - функция системы, набор действий, или элементарное действие. На IDEF0 диаграммах изображается пря­моугольником. Детализируется при помощи диаграмм нижних уровней. Диаграмма главного процесса представлена на рисунке 1.

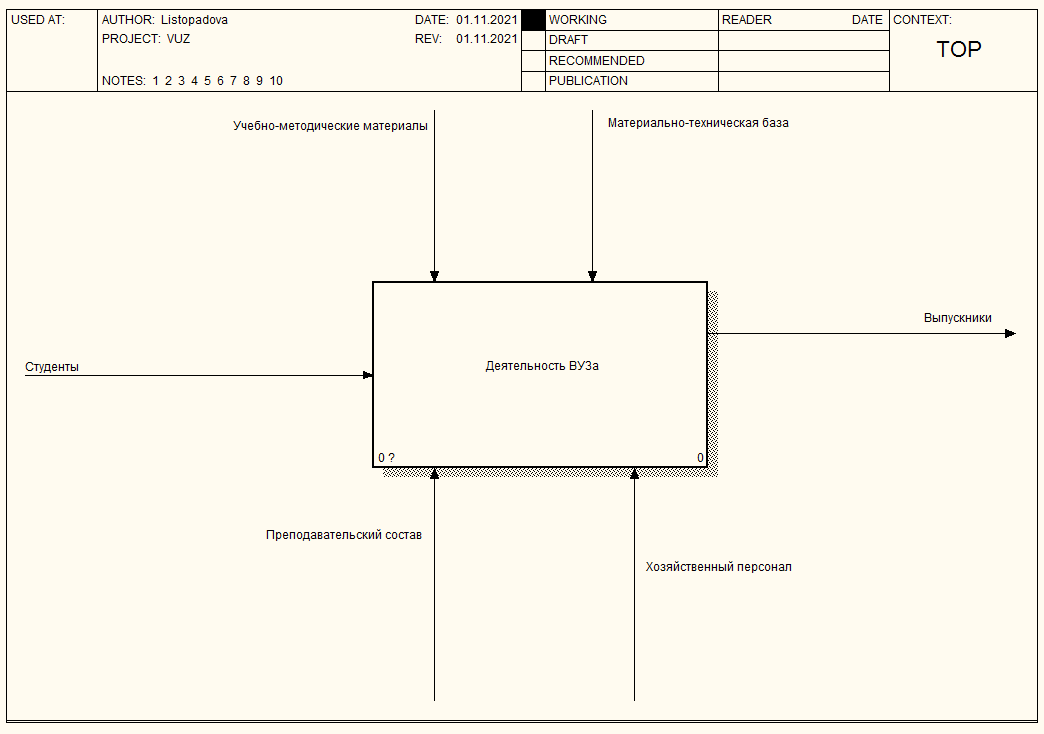


Рисунок 1. Контекстная диаграмма главного процесса.

Составим декомпозицию главного рабочего процесса, он состоит из шести основных процессов: Прием абитуриентов, Теоретическое обучение, Прочие виды деятельности, Производственная практика, Государственные экзамены и Организация хоздеятельности предприятия.

Диаграмма декомпозиции главного процесса представлена на рисунке 2.

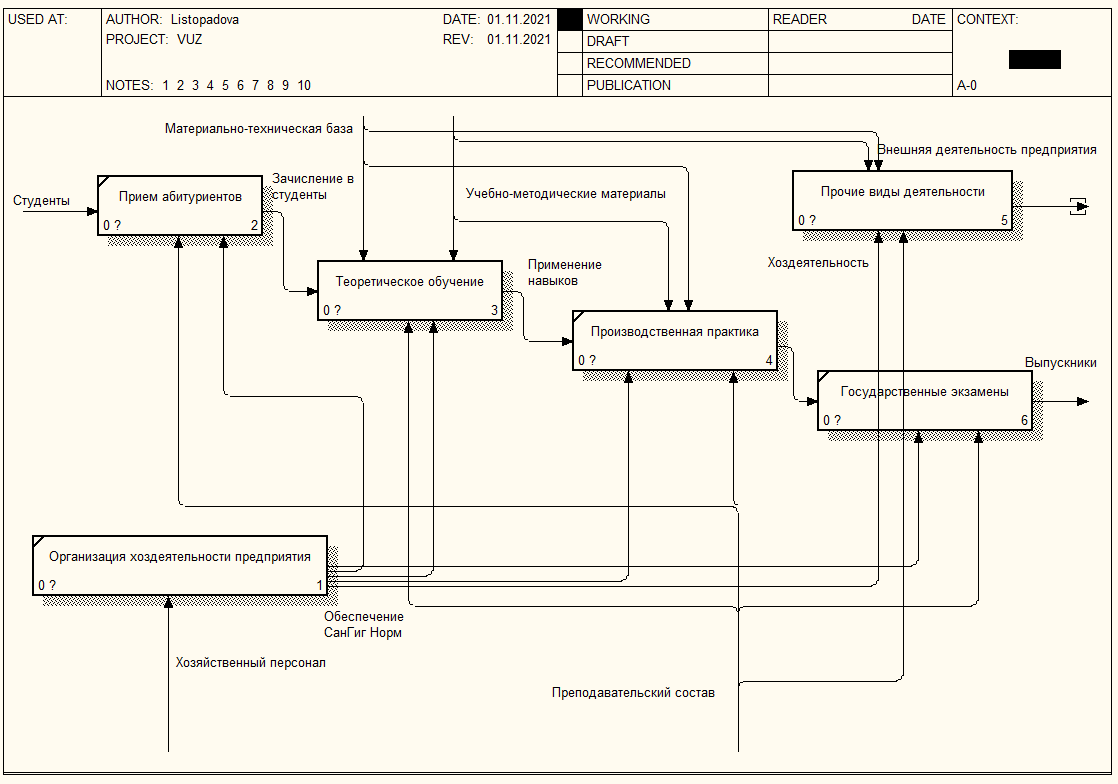


Рисунок 2. Диаграмма декомпозиции главного процесса.

Каждый из этих процессов может быть так же подвергнут декомпозиции. Теоретическое обучение состоит из нескольких технологических процессов. Декомпозиция процесса представлена на рисунке 3.

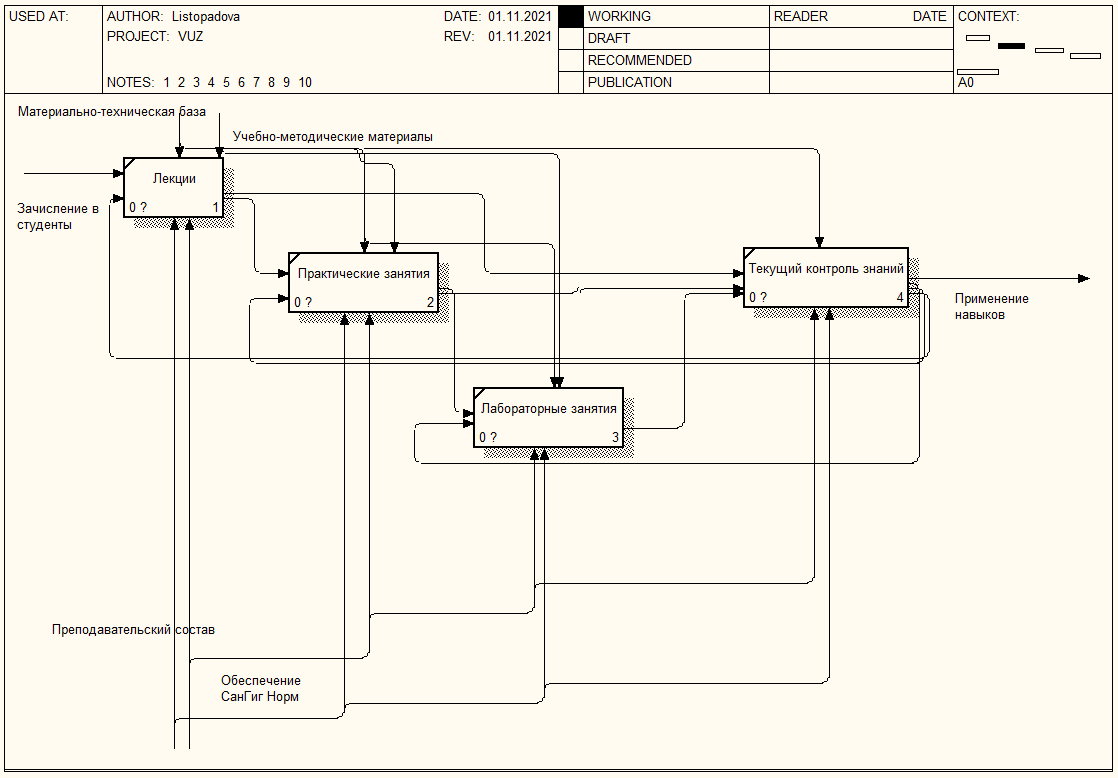


Рисунок 3. Декомпозиция процесса «Теоретическое обучение».

Декомпозиция процесса «Прочие виды деятельности» представлена на рисунке 4.

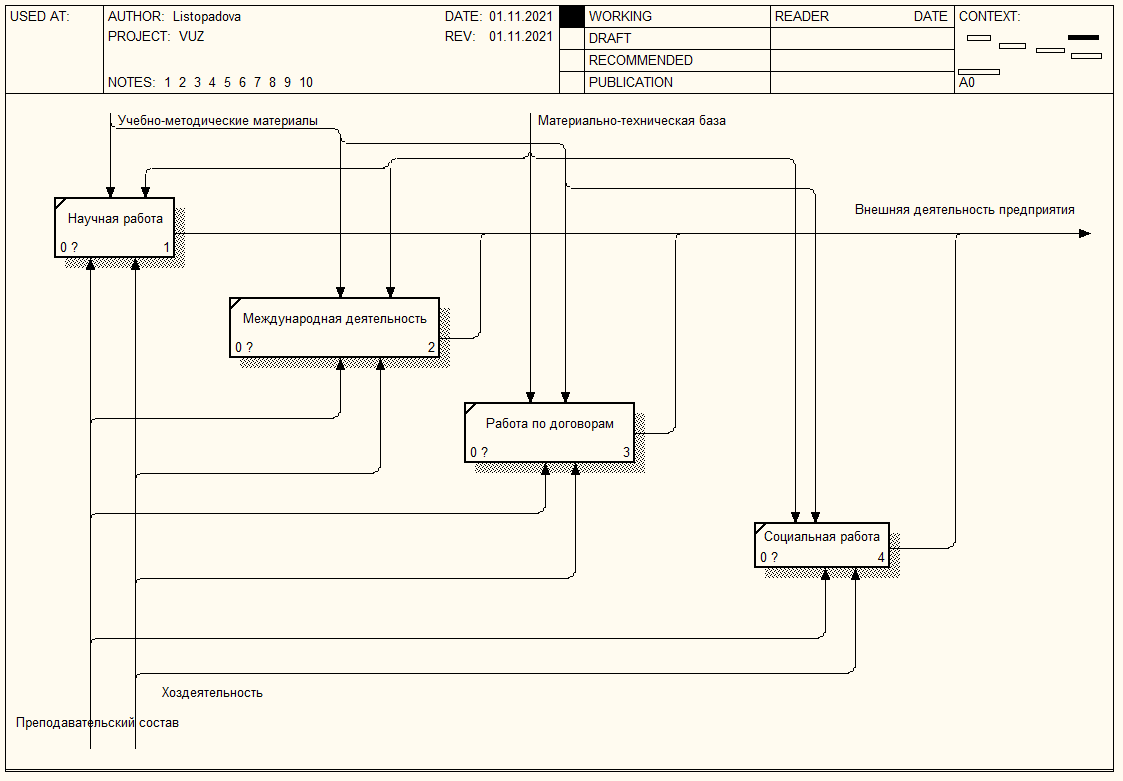


Рисунок 4. Декомпозиция процесса «Прочие виды деятельности».

# 49 Электронная картотека.

# Читатель может ввести: название книги и получить ФИО автора и специальные данные книги; область знаний и получает названия книг, ФИО авторов, специальные данные книги; автора и получает область знаний, название книги. Результаты поиска можно распечатать. Администратор вводит новые книги, то есть название, автора, специальные данные и область знаний новой книги.

Создание контекстной диаграммы.

Контекстная диаграмма отражает отношение системы с внешней средой. Она должна содержать только один процесс, называемый общей фразой, обозначающий в целом деятельность всей моделируемой системы. В данном случае это процесс будет называться «Обработать данные».

Процесс - функция системы, набор действий, или элементарное действие. В названии процесса обязательно должен присутствовать глагол. Обозначается прямоугольником с закругленными углами. Детализируется при помощи диаграмм нижних уровней.

На контекстной диаграмме изображены все внешние сущности, то есть объекты, поставляющие информацию в систему или получающие ее. В данном примере их две - читатель и администратор.

Потоки данных между процессом и внешними сущностями отображают получаемую и передаваемую информацию.

От администратора к процессу идет один поток данных - Данные о новых книгах. Это новая информация для базы данных.

От пользователя к процессу идут три потока данных - Название книги, Область знаний и автор. Эти потоки необходимы для получения нужных пользователю сведений.

От процесса к пользователю идут три потока данных – Область знаний и название книги, Названия книг, ФИО авторов и специальные данные, и ФИО автора и специальные данные книги. Эти потоки соответствуют данным, выдаваемым на монитор компьютера и отчетам, напечатанным на принтере.

От процесса к администратору идет один поток данных – Данные о пользователях.

Контекстная диаграмма приведена на рисунке 5.

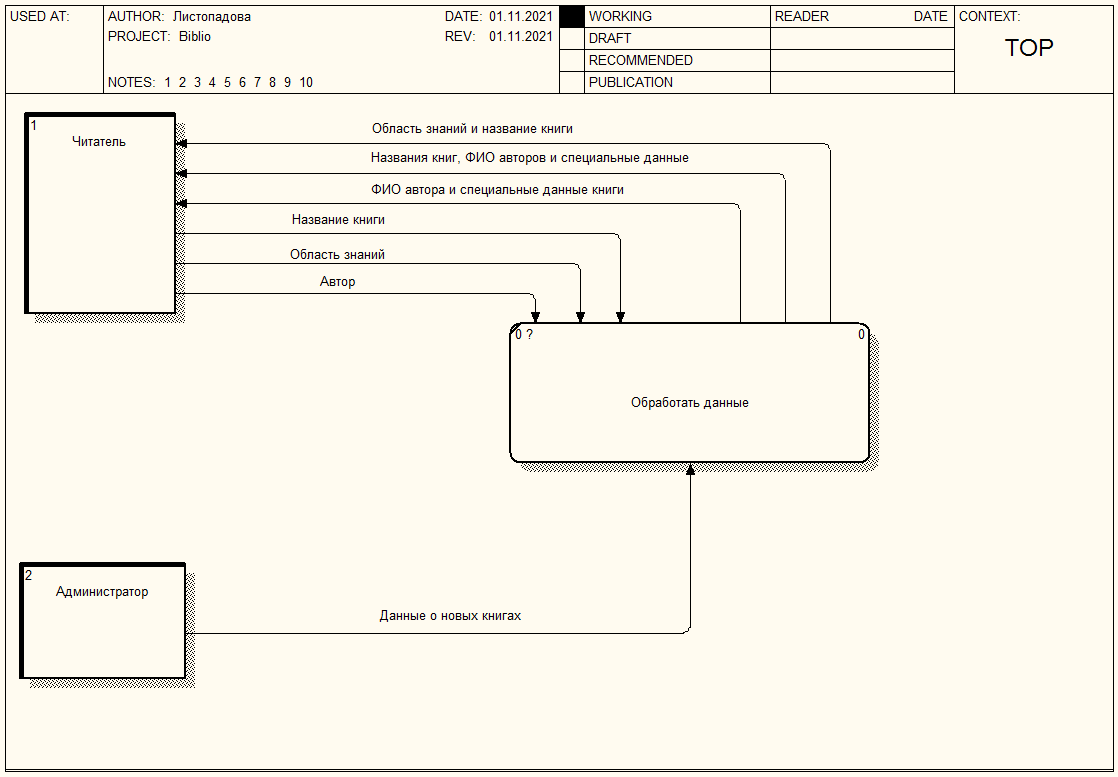


Рисунок 5. Контекстная диаграмма системы.

На диаграмме первого уровня необходимо разместить шесть процессов, соответствующие основным функциям справочной системы.

Эти процессы можно назвать так:

1 Процесс 5 - «Внести новые данные о книгах» - носит в базу данных сведения вводимые администратором.

2 Процесс 2 - «Найти область знаний и название книги» - выполняет поиск область знаний и названия книги по её автору.

3 Процесс 3 - «Найти названия книг и ФИО авторов» - по введенной знаку области знаний выполняет поиск и выдает названия книг и их авторов.

4 Процесс 4 - «Найти ФИО и специальные данные» - по введенному названию книги выполняет поиск и выдает ФИО автора и специальные данные книги.

5 Процесс 6 - «Вывести информацию на экран» - выводит для полученную информацию на экран.

6 Процесс 1 - «Вывести информацию на принтер» - выводит для информацию на печатающее устройство.

Потоки на контекстной диаграмме направленные от процесса к пользователю на диаграмме первого уровня направлены стрелкой вовне, то есть от процессов. В дальнейшем они будут обозначаться как исходящие потоки. Потоки, направленные на контекстной диаграмме от пользователя к процессу, на диаграмме первого уровня направлены стрелкой к процессам. В дальнейшем они будут обозначаться как входящие потоки.

Чтобы получить данные процесс должен направить к базе данных запрос. Из базы данных он получает результат поиска. Надо создать два новых потока. Один от базы данных к процессу, второй от процесса к базе данных.

Аналогичным образом потоки данных распределяются между остальными процессами.

Диаграмма первого уровня представлена на рисунке 6.

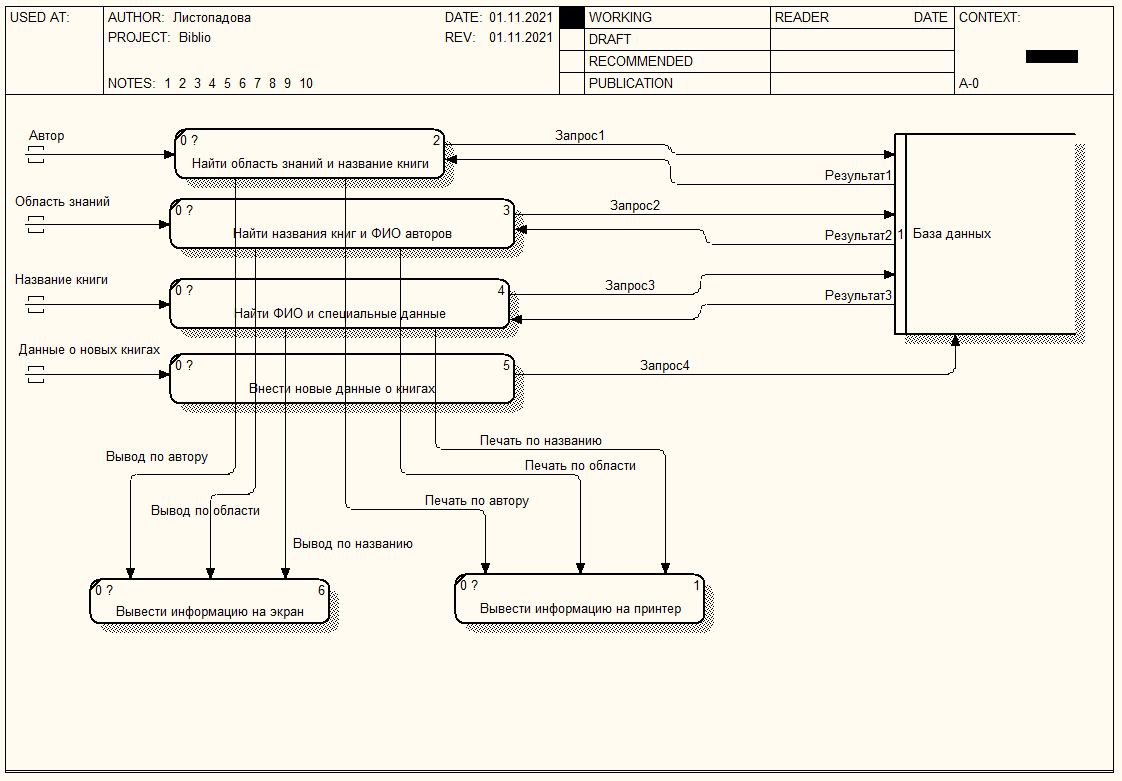


Рисунок 6. Диаграмма первого уровня.

На диаграмме второго уровня будет три процесса. Эти процессы детализируют работу процесса «Найти область знаний и название книги». Названия этих процессов будут следующими:

«Сформировать запрос» - получает поток «Автор», выдает поток данных, направленный к процессу 2 «Направить запрос в базу данных».

«Направить запрос в базу данных» - получает поток данных от процесса 1, и направляет его в базу данных как Запрос1.

«Сформировать отчет» - получает поток данных «Результат1» и выдает потоки для вывода данных на экран и печать.

Диаграмма второго уровня представлена на рисунке 7.

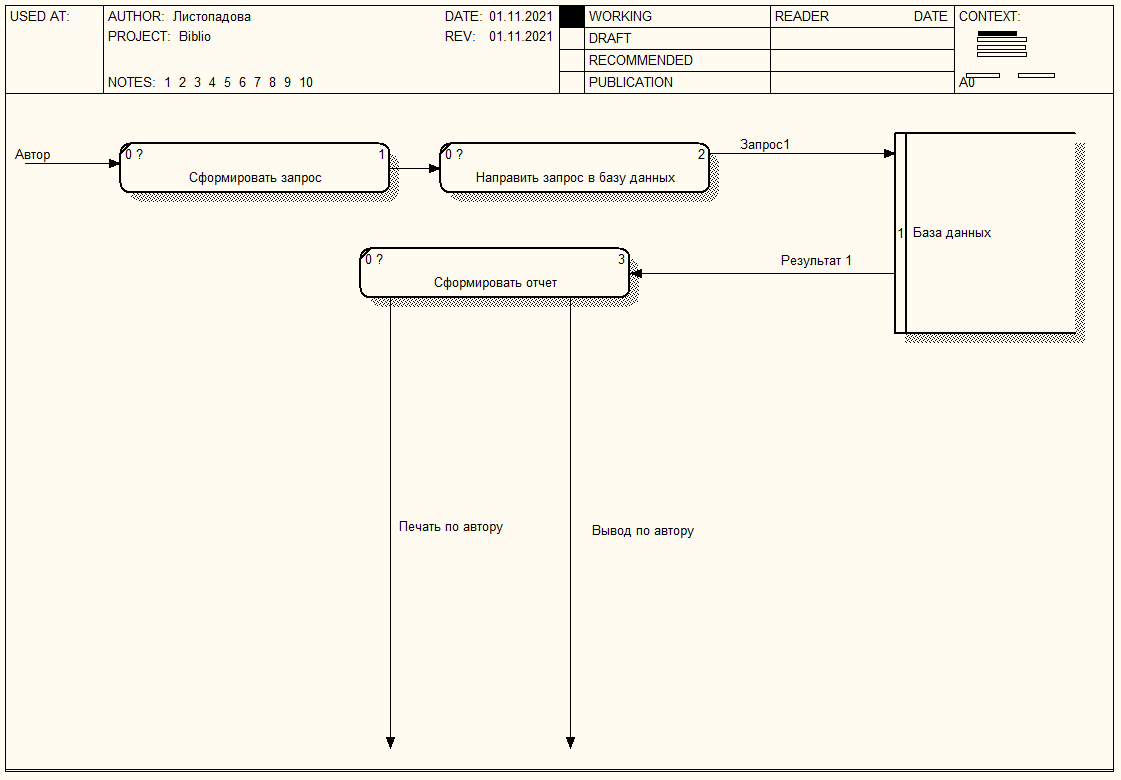


Рисунок 7. Диаграмма второго уровня.

# 59 База данных «Посещение кружков»

# Содержит информацию о школьниках (ФИО, школа, класс, адрес, телефон), о кружках (название, руководитель, число занятий в неделю, оплата за месяц), о посещении (ФИО, школа, класс, название, руково-дитель, дата записи). Один ребенок может посещать один и более кружков, один кружок должен состоять хотя бы из одного ребенка.

Создадим три сущности: Школьники, Кружки и Посещение, в которых будем хранить соответствующие им атрибуты.

Сущность Школьники будет иметь следующие атрибуты:

* ФИО,
* школа,
* класс,
* адрес,
* телефон.

Сущность Кружки будет иметь атрибуты:

* название,
* руководитель,
* число занятий в неделю,
* оплата за месяц.

Сущность Посещение по условию обладает атрибутами:

* ФИО,
* школа,
* класс,
* название,
* руководитель,
* дата записи.

Можно видеть, что после организации связей между сущностями у нас образуется избыточное дублирование атрибутов: Школа, Класс – они есть в сущности Школьники, и атрибута Руководитель – он есть в сущности Кружки. Оставим сущность Посещение в виде:

* ФИО,
* название,
* дата записи.

Для создания сущности «Школьники» откроем диалоговое окно создания сущности, внешний вид которого приведен на рисунке 8.

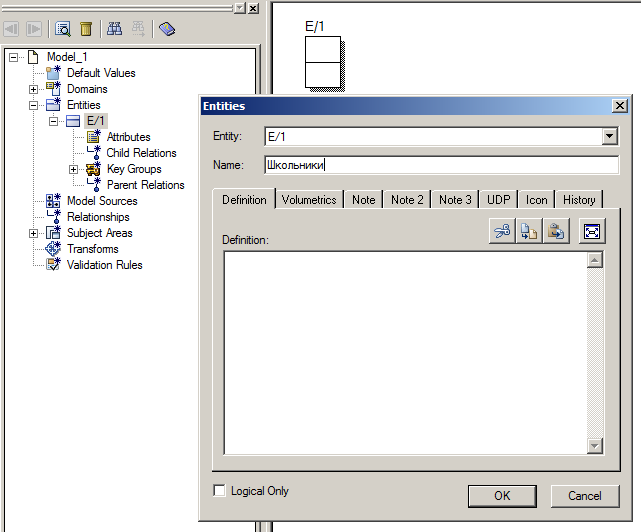


Рисунок 8. Окно создания сущности Школьники.

Создадим сущность и укажем присущие ей атрибуты и их тип. Зададим первичный ключ. Образец окна редактирования атрибутов сущности представлен на рисунке 9.

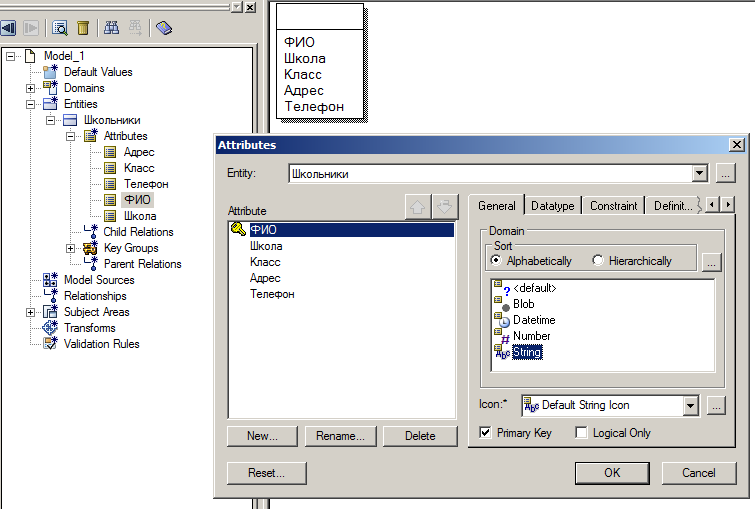


Рисунок 9. Окно редактирования атрибутов сущности.

Созданные сущности и их атрибуты представлены на рисунке 10.

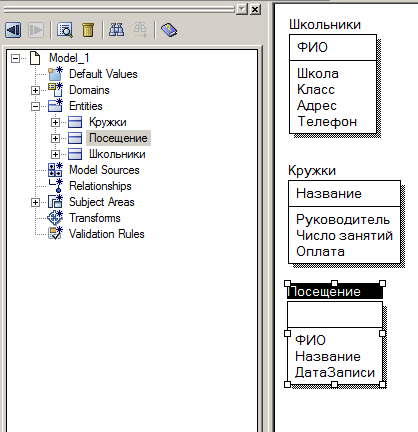


Рисунок 10. Сущности базы данных.

После создания сущностей и добавления их атрибутов необходимо установить между ними связи. Связи между сущностями базы данных представлены на рисунке 11.

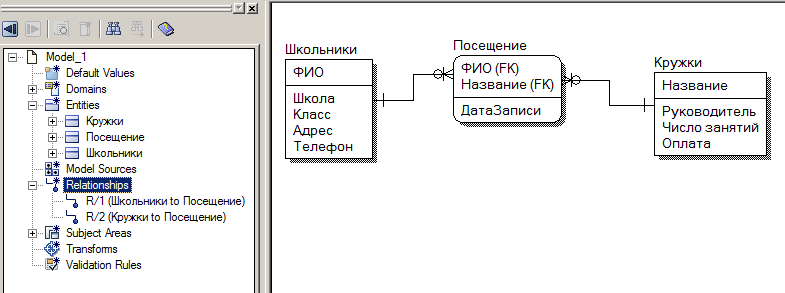


Рисунок 11. Связи сущностей в базе данных.

**87 Разрабатывается программная система учета товаров в продовольственном магазине. Согласно собранным сведениям, в системе должны работать продавец и заведующий магазином. С помощью системы продавец должен выполнять следующие функции: внести поставки продуктов, внести остатки продуктов на конец дня.**

**С помощью системы заведующий магазином просматривает данные о продуктах с истекшим сроком годности и создает отчет о списании просроченных продуктов. Получает отчеты об остатках всех продуктов.**

**Предусмотрите идентификацию пользователей и защиту паролями.**

Так как в магазине работает заведующий и продавец, необходима регистрация в системе.

Для построения диаграммы вариантов использования выделены следующие исполнители: Продавец и Заведующий.

Для Продавца выделены следующие варианты использования:

* Идентифицироваться в системе;
* Внести поставки продуктов;
* Внести остатки продуктов;

Для Заведующего выделены следующие варианты использования:

* Идентифицироваться в системе;
* Получить отчет об остатках продуктов;
* Получить отчет о просроченных продуктах;
* Создать отчет о списании продуктов.

Разработанная диаграмма вариантов использования представлена на рисунке 12.

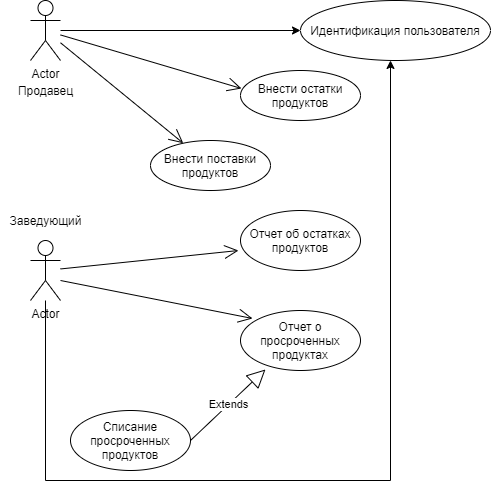
****

Рисунок 12. Диаграмма вариантов использования

Для решения задачи автоматизации учета продаж были разработаны 3 таблицы.

Диаграмма классов представлена на рисунке 13. На диаграмме выделены для каждой таблицы атрибуты и операции, которые возможно осуществить со справочником. Указаны поля, созданы связи и указан их тип.

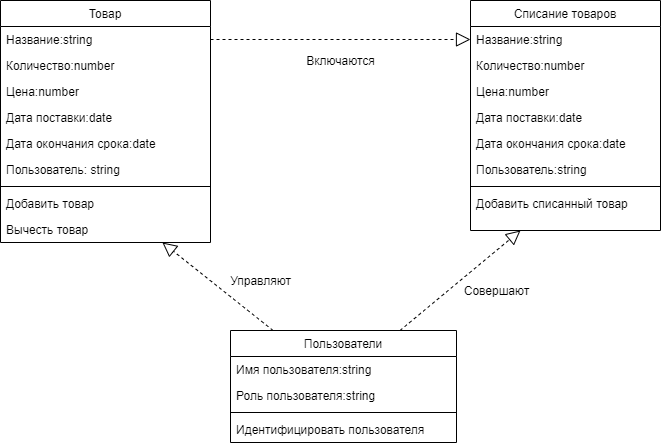


Рисунок 13. Диаграмма классов.

**Список используемых источников**

1 ГОСТ 19.201-78. ЕСПД. Техническое задание. Требования к содержанию и оформлению.

2 ГОСТ 19.504-79.ЕСПД. Руководство программиста.

3 Бек, К. Экстремальное программирование / К. Бек; пер. с англ. -СПб., 2003.

4 Буч, Г. Язык UML. руководство пользователя / Г.Буч, Д.Рамбо, А.Джекобсон; пер. с англ. - М.:, 2000.

5 Липаев, В.В. Системное проектирование сложных программных средств для информационных систем / В.В. Липаев. - М., 1999.

6 Маклаков, С.В. BPwin и ERwin. CASE-средства разработки информационных систем / В.В.Липаев. - М.:,1999.

7 Мартин, Р. Принципы, паттерны и методики гибкой разработки на языке C# / Р.Мартин, М.Мартин; пер. с англ. - СПб., 2009.