Федеральное государственное бюджетное образовательное учреждение

высшего образования

**Кафедра «Цифровые технологии и моделирование»**

**Структурное и объектно-ориентированное моделирование   
процессов и систем**

Учебно-методические указания

для выполнения курсовой работы студентами направления подготовки

09.03.03 Прикладная информатика

**Уфа**

**2018**

Приведены методические указания для выполнения курсовой работы по дисциплине «Программная инженерия», включающие описание заданий по разработке приложений и требований по оформлению пояснительной записки.

Предназначено для подготовки бакалавров всех форм обучения по направлению подготовки 09.03.03 Прикладная информатика, при изучении дисциплины «Программная инженерия».

**Оглавление**

[Тема курсовой работы 4](#_Toc513365240)

[Варианты заданий 4](#_Toc513365241)

[Требования к оформлению пояснительной записки 5](#_Toc513365242)

[Порядок защиты работы и контрольные вопросы 6](#_Toc513365243)

[Защита курсовой работы и ее оценивание 6](#_Toc513365244)

[Контрольные вопросы при защите работы 6](#_Toc513365245)

[Приложение А. Теоретические основы 7](#_Toc513365246)

[Диаграммы UML 7](#_Toc513365247)

[Используемые обозначения и определения 7](#_Toc513365248)

[Пример использования обозначений UML 10](#_Toc513365249)

[Диаграммы вариантов использования 10](#_Toc513365250)

[Диаграммы последовательности 11](#_Toc513365251)

[Диаграммы коопераций (обмена данными) 13](#_Toc513365252)

[Диаграммы классов 14](#_Toc513365253)

[Диаграмма деятельности 15](#_Toc513365254)

[Диаграммы состояний 16](#_Toc513365255)

[Примеры реализации диаграмм в программном коде 17](#_Toc513365256)

[Диаграмма классов 17](#_Toc513365257)

[Диаграммы последовательности и коопераций 17](#_Toc513365258)

[Диаграммы деятельности 18](#_Toc513365259)

[Диаграммы состояний 18](#_Toc513365260)

[Приложение Б. Инструменты для рисования UML-диаграмм 20](#_Toc513365261)

[Приложение В. Лист выдачи задания на курсовую работу 21](#_Toc513365262)

[Приложение Г. Титульный лист курсовой работы 22](#_Toc513365263)

[Список литературы 23](#_Toc513365264)

# Тема курсовой работы

Тема: **Структурное и объектно-ориентированное моделирование процессов и систем.**

Необходимо с использованием *Унифицированного языка моделирования* (Unified Modeling Language, ***UML***) для заданной системы (в соответствии с вариантом задания ) разработать несколько типов визуальных диаграмм:

1. диаграммы вариантов использования;
2. диаграммы взаимодействия;
   1. диаграммы последовательности;
   2. кооперативные диаграммы;
3. диаграммы классов;
4. диаграммы деятельности;
5. диаграммы состояний;

Для каждой из диаграмм сделать описание ее назначения, спецификацию и показать использование в разработке программного обеспечения (разработке классов, программного кода, документирования системы).

## Варианты заданий

1. Моделирование программной системы облуживания банкомата.
2. Моделирование программной системы управления работой светофоров на перекрестке.
3. Моделирование программной системы работы лифтов здания.
4. Моделирование информационной системы учета товаров на складах.
5. Моделирование информационной системы учета сотрудников в отделе кадров.
6. Моделирование информационной системы учета студентов в системе деканата.
7. Моделирование программной системы тестирования знаний.
8. Моделирование информационной системы учета выдачи книг в библиотеке.
9. Моделирование программной системы дистанционного обучения.
10. Моделирование информационной системы материального учета компьютеров.

# Требования к оформлению пояснительной записки

Пояснительная записка (ПЗ) по работе должна быть набрана на компьютере в текстовом процессоре MS Word, распечатана, сброшюрована и содержать следующие разделы:

1. Титульный лист.
2. Задание на курсовую работу.
3. Содержание.
4. Отчет по курсовой работе.
5. Выводы по работе.
6. Список литературы.

На *титульном листе* указывается название учебного заведения, кафедра, название дисциплины, вид работы, тема работы, руководитель, исполнитель, дата и т.д. Пример см. Приложение 2.

*Задание на* курсовую *работу*. Приводится текст варианта задания.

*Содержание*. В содержании указывается наименования разделов отчета и номера страниц, с которых начинается соответствующий раздел.

*Отчет по* курсовой работе. Описываются возможности ***UML*** для решения поставленной задачи. Приводится постановка задачи, описание диаграмм, их назначения, спецификация и использование в разработке программного обеспечения.

*Выводы по работе*. Дается оценка полученным результатам.

*Список литературы*. Приводится список литературных источников, использованных в процессе подготовки и выполнения работы.

*Объем* пояснительной записки по курсовой работе не менее 20 страниц печатного текста формата А4. Поля - 20 мм. Шрифт - 12пт, межстрочный интервал – одинарный.

# Порядок защиты работы и контрольные вопросы

## Защита курсовой работы и ее оценивание

Разработанная программа и оформленная пояснительная записка должны быть представлены к указанному и утвержденному сроку. После проверки ПЗ и работы приложения она рекомендуется к защите.

Итоговая оценка складывается из нескольких составляющих: срока выполнения, качества и работоспособности программы, качества оформления ПЗ и результатов защиты.

При этом:

1. Не позднее недели после выдачи задания на курсовую работу надо разобраться в поставленной задаче, предложить свой вариант ее реализации и интерфейс программы, согласовать его с руководителем курсовой работы.

2. Если программа и пояснительная записка не будут представлены к утвержденному сроку, оценка может быть снижена на 1 балл.

*Пояснительная записка* по курсовой работе должна быть представлена преподавателю в распечатанном виде (подшитый бумажный документ) и в электронном виде.

*Исходный код* приложения представляется в *электронном виде*.

## Контрольные вопросы при защите работы

1. Перечислить диаграммы ***UML.***
2. Объяснить назначение одной из диаграмм ***UML***.
3. Объяснить понятие ***Сущности*** в ***UML***. Привести примеры обозначений ***Сущности*** в ***UML***.
4. Объяснить понятие ***Связи*** в ***UML***. Привести примеры обозначений ***Связи*** в ***UML***.
5. Объяснить понятие ***Диаграммы*** в ***UML***. Привести примеры обозначений ***Диаграммы*** в ***UML***.
6. Описать обозначение класса в ***UML.***
7. Описать обозначение интерфейса в ***UML.***
8. Описать обозначение автомата в ***UML.***
9. Описать обозначение ассоцияции в ***UML.***
10. Описать обозначение агрегации в ***UML.***
11. Описать обозначение композиции в ***UML.***

# Приложение А. Теоретические основы

# Диаграммы UML

Унифицированный язык моделирования (Unified Modeling Language, ***UML***) – это графический язык для визуализации, специфицирования, конструирования и документирования систем.

***UML*** позволяет создавать несколько типов визуальных диаграмм, например:

1. диаграммы вариантов использования;
2. диаграммы взаимодействия;
   1. диаграммы последовательности;
   2. кооперативные диаграммы;
3. диаграммы классов;
4. диаграммы деятельности;
5. диаграммы состояний.

Построение диаграмм используется для иллюстрации различных аспектов системы (назначение и решаемые задачи, функциональность, используемые классы и их взаимодействие, процессы и т.п.) в соответствии с поставленными целями. Далее показано назначение и примеры диаграмм по книгам [1, 2].

## Используемые обозначения и определения

Словарь ***UML*** включает три специфицированных вида:

1. Сущности.
2. Связи.
3. Диаграммы.

*Сущности* – это абстракции, которые являются основными элементами модели, *связи* (relationships) соединяют их между собой, а *диаграммы* группируют представляющие интерес наборы сущностей.

*Сущности* – это «имена существительные» в UML. Они включают в себя интерфейсы, классы, кооперации, варианты использования, активные классы, компоненты и узлы.

*Интерфейс*, представляемый классом, изображается в виде маленького круга, соединенного линией с рамкой класса, либо в виде диаграммы реализации класса. Интерфейс, запрашиваемый классом от некоторого другого класса, представлен маленьким полукругом, соединенным с рамкой класса линией, как показано на Рисунке 1.

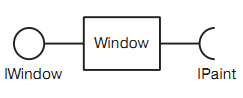
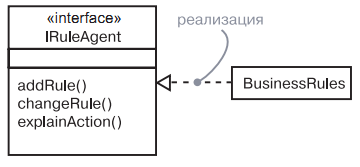
 

Рисунок 1

Пример *моделирования* *класса* показан на Рисунке 2.

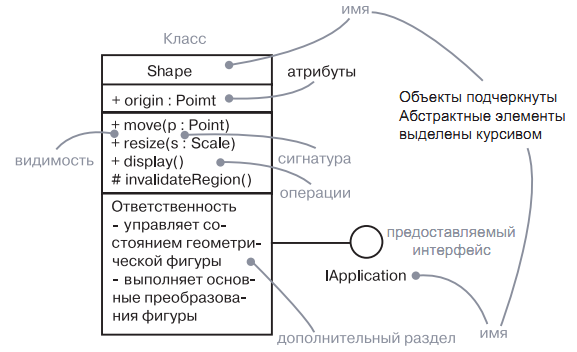


Рисунок 2

Поведенческие сущности – это динамические части ***UML***. Они включают взаимодействия и автоматы. Пример *диаграммы автомата* показан на Рисунке 3.

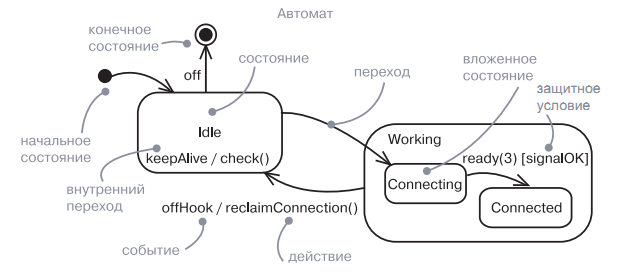


Рисунок 3

***Зависимость*** – это семантическая связь между двумя сущностями, при которой изменение одной сущности (независимой) может повлиять на семантику другой (зависимой), Рисунок 4.

***Обобщение*** – связь, при которой объекты специализированного элемента (потомка) могут замещать объекты более общего элемента (родителя), Рисунок 4.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Рисунок 4

***Реализация*** - семантическая связь между классификаторами, когда один из них специфицирует соглашение, которого второй обязан придерживаться, Рисунок 5.



Рисунок 5

***Ассоциация*** – структурная связь, которая описывает набор ссылок. Ссылка – это экземпляр ассоциации, представляющий собой соединение объектов, Рисунок 6. Ассоциация характерна для связи объектов, семантически реализованных в таблицах базы данных.



Рисунок 6

***Агрегация*** – это особый вид ассоциации, подразумевает, что объект-целое обладает объектами-частями, Рисунок 7.

***Композиция*** – это форма агрегации с четко выраженными отношениями владения и совпадением времени жизни частей и целого, Рисунок 7.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Рисунок 7

## Пример использования обозначений UML

Пример диаграммы классов и связей показан на Рисунке 8.

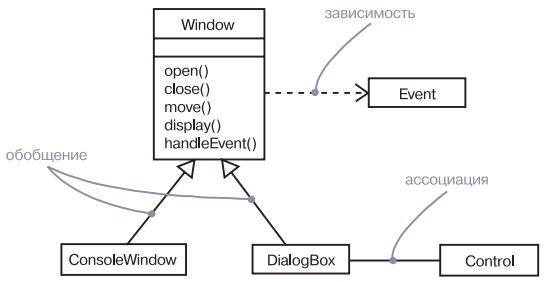


Рисунок 8

## Диаграммы вариантов использования

Диаграммы вариантов использования отображают *взаимодействие* между вариантами использования, представляющими функции системы, и актерами, представляющими людей или системы, получающие или передающие информацию в этой системе. Таким образом, диаграммы вариантов использования содержат субъект, варианты использования, действующие лица, а также связи зависимости, обобщения, ассоциации, и могут содержать примечания и ограничения.

Пример диаграммы вариантов использования для *банковского автомата* (ATM) показан на Рисунке 9.

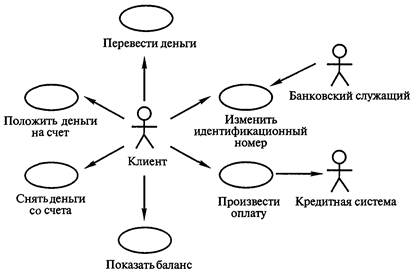


Рисунок 9

Диаграмма вариантов использования *отражает требования к функциональности системы с точки зрения пользователя,* т.е. чтосистема должна делать. Диаграмма также показывает, какие действующие лица инициируют варианты использования. В данном примере имеются следующие варианты использования (функции, выполняемые системой):

* Снять деньги со счета;
* Перевести деньги;
* Положить деньги на счета;
* Показать баланс;
* Изменить идентификационный номер;
* Произвести оплату.

Стрелка, направленная от действующих лиц показывает какие варианты использования инициируют эти лица.

Стрелка, направленная от варианта использования к действующему лицу, показывает, что вариант использования предоставляет некоторую информацию действующему лицу.

В данном случае вариант использования ***Произвести оплату*** предоставляет ***Кредитной системе*** информацию об оплате по кредитной карточке.

Наиболее общий случай применения вариантов использования – это моделирование поведения системы, подсистемы или класса с целью сосредоточить внимание на том, что делается, а не как это делается. Это позволяет:

* экспертам в предметной области показать внешнее представление системы понятное разработчикам для конструирования его внутреннего представления и конечным пользователям для понимания функциональности системы и ее использования,
* смоделировать требования к системе,
* показать базу для тестирования системы в процессе его разработки.

## Диаграммы последовательности

Диаграммы последовательности отражают *поток событий*, происходящих *в рамках варианта использования.* Например, вариант использования ***Снять деньги*** предусматривает несколько возможных последовательностей:

* снятие денег,
* попытка снять деньги при отсутствии их достаточного количества на счету,
* попытка снять деньги по неправильному идентификационному номеру
* и некоторые другие.

Далее показан *нормальный сценарий* снятия денег со счета (при отсутствии проблем), Рисунке 10.

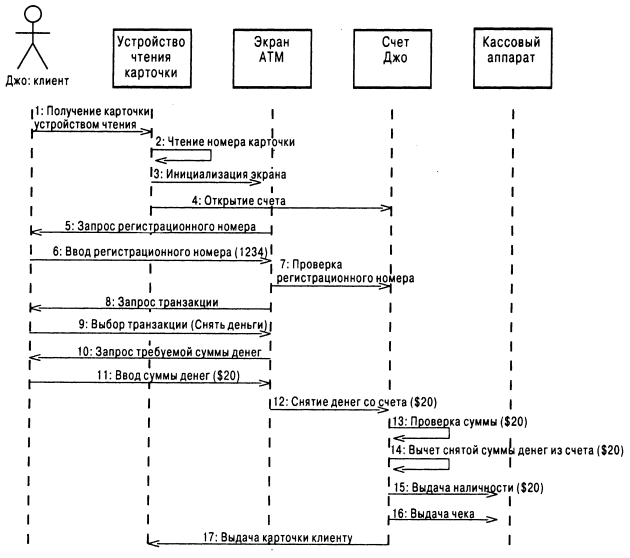


Рисунок 10

В верхней части диаграммы показаны *все действующие лица и объекты*, требуемые системе для выполнения варианта использования ***Снять деньги***.

Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций. Следует отметить также, что *на диаграмме последовательности показаны именно объекты*, а не классы.

Основное содержимое диаграммы последовательности – *сообщения*. Они изображаются стрелками, направленными от одной линии жизни к другой. Стрелка указывает на приемник сообщения. Если сообщение асинхронно, то рисуется стрелка, а если синхронно (вызов), то закрашенная стрелка. Ответ на синхронное сообщение (возврат из вызова) показывается пунктирной стрелкой. Сообщение возврата может быть опущено, поскольку каждый вызов неявно подразумевает возврат.

Изучая диаграмму последовательности, пользователи знакомятся со спецификой своей работы. Аналитики видят последовательность (поток) действий, разработчики — объекты, которые надо создать, и их операции. Специалисты по контролю качества поймут детали процесса и смогут разработать тесты для их проверки.

## Диаграммы коопераций (обмена данными)

Диаграммы коопераций отражают ту же самую информацию, что и диаграммы последовательности. Однако делают они это по-другому и с другими целями. Если диаграмма последовательности показывает взаимодействие между действующими лицами и объектами во времени, то *на диаграмме коопераций связь со временем отсутствует*. Так, можно видеть, что устройство для чтения карточки выдает счету Джо инструкцию открыться, а счет Джо заставляет это устройство вернуть карточку владельцу. Непосредственно *взаимодействующие объекты соединены линиями*. Если, например, устройство для чтения карточки общается непосредственно с экраном ATM, между ними следует провести линию. Отсутствие линии означает, что непосредственное сообщение между объектами отсутствует.



Рисунок 11

Специалисты по контролю качества и архитекторы системы смогут понять распределение процессов между объектами. Допустим, что какая-то диаграмма коопераций напоминает звезду, где несколько объектов связаны с одним центральным объектом. Архитектор системы может сделать вывод, что система слишком сильно зависит от центрального объекта, и необходимо перепроектировать ее для более равномерного распределения процессов. На диаграмме последовательности такой тип взаимодействия было бы трудно увидеть.

Поскольку и диаграммы последовательности, и диаграммы коопераций (коммуникации) наследуют одну и ту же информацию метамодели ***UML***, то они семантически эквивалентны. В результате вы можете взять одну диаграмму и преобразовать ее в другую, без каких бы то ни было потерь информации.

Диаграмма коопераций наглядно показывает, как связаны объекты, а диаграмма последовательности - направление и последовательность сообщений.

## Диаграммы классов

Диаграммы классов отражают *взаимодействие между классами* системы. Классы содержат данные и методы. Так, класс ***Счет*** содержит идентификационный номер клиента, пин-код и баланс, а также методы работы со счетом. На диаграмме классов *класс создается для каждого типа объектов* из диаграмм *последовательности* или диаграмм *коопераций*.

На диаграмме классов для варианта использования ***Снять деньги*** показаны связи между классами, Рисунок 12. В этом процессе задействованы четыре класса: ***Card Reader*** (устройство для чтения карточек), ***Account*** (счет), ***ATM*** (экран ATM) и ***Cash Dispenser*** (кассовый аппарат).

Каждый класс на диаграмме классов изображается в виде прямоугольника, разделенного на три части. В верхней части указывается имя класса, затем его *атрибуты и далее методы.*

Атрибут — это некоторая информация, характеризующая класс. Например, класс ***Account*** (счет) имеет три атрибута: ***Account Number*** (номер счета), ***PIN*** (пин-код) и ***Balance*** (баланс).

Методы класса, отражают его *поведение.*

Связывающие классы линии показывают взаимодействие между классами.

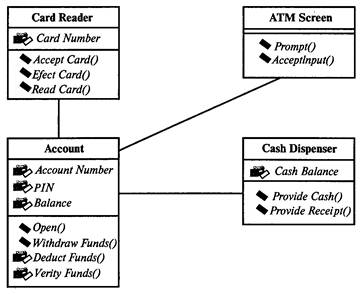


Рисунок 12

Разработчики используют диаграммы классов для реального создания классов. Такие инструменты, как ***Rose***, генерируют основу кода классов, которую программисты заполняют деталями на выбранном ими языке. С помощью этих диаграмм аналитики могут показать детали системы, а архитекторы — понять ее проект. Если, например, какой-либо класс несет слишком большую функциональную нагрузку, это будет видно на диаграмме классов, и архитектор сможет перераспределить ее между другими классами. С помощью диаграммы можно также выявить случаи, когда между сообщающимися классами не определено никаких связей. Диаграммы классов следует создавать, чтобы показать взаимодействующие классы в каждом варианте использования. Можно строить также более общие диаграммы, охватывающие все системы или подсистемы.

## Диаграмма деятельности

**Диаграмма деятельности** позволяет описать логику процедур, бизнес-процессы и потоки работ. Во многих случаях они напоминают блок-схемы, но поддерживают параллельное процессы. На Рисунке 13 показан пример простой ***диаграммы деятельности***.

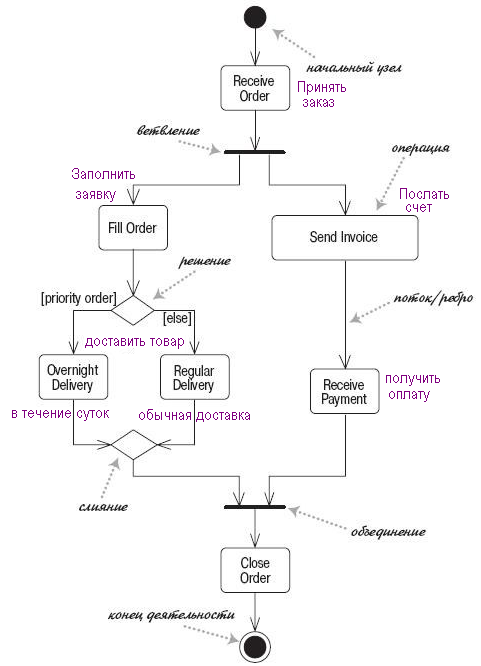


Рисунок 13

После принятия заказа идет ветвление, которое имеет два параллельных потока - операции Заполнить заявку и Послать счет, а также последующие за ними действия выполняются параллельно, что означает, что последовательность операций в разных потоках не согласована по времени. Можно заполнить заявку, послать счет, доставить товар, а затем получить оплату, или можно послать счет, получить оплату, заполнить заявку, а затем доставить товар.

При наличии параллелизма необходима конечная синхронизация. Мы не закрываем заказ, пока он не оплачен и не доставлен.

Использование диаграмм деятельности для изображения блок-схемы операции не обязательно в ***UML***. Более целесообразно написание тела операции на определенном языке программирования. Есть смысл прибегать к диаграммам деятельности для моделирования операций, поведение которых трудно для понимания на основе одного лишь исходного кода.

## Диаграммы состояний

При объектной декомпозиции система разбивается на объекты, которые взаимодействуют друг с другом, обмениваясь сообщениями. Сообщения  описывают или представляют собой некоторые события. После получение объектом сообщения выполняется процедура его обработки. При таком подходе система становится событийно управляемой, поэтому разработчикам  необходимо знать, как должен реагировать объект на определенные события. Инициаторами событий могут быть как объекты самой системы, так и её внешнее окружение.

Описать поведение объекта помогает *диаграмма состояний*. Также зачастую диаграмма состояний используется аналитиками для описания последовательности переходов объекта из одного состояния в другое. Таким образом, диаграммы состояний предназначены для *моделирования различных состояний, в которых может находиться объект*. В то время как диаграмма классов показывает статическую картину классов и их связей, диаграммы состояний применяются при описании динамики поведения системы.

Основными элементами диаграммы состояний являются ***Состояние*** и ***Переход***. На диаграмме состояний *деятельность символизирует* ***состояние****, в котором объект находится продолжительное количество времени, в то время как действие моментально*.

***Деятельность*** – текущий процесс внутри автомата.

***Действие*** – исполняемое вычисление, в результате которого изменяется состояние модели или возвращается некоторое значение. Изображается в виде прямоугольника с закругленными углами.

***Переход*** – связь между двумя состояниями, означающая, что объект в первом состоянии должен выполнить определенные действия и перейти во второе состояние, когда произойдет определенное событие и будут удовлетворены заданные условия.

Так, банковский счет может иметь несколько различных состояний:

* открыт,
* закрыт,
* превышен кредит.

Поведение счета меняется в зависимости от состояния, в котором он находится. Переход может быть инициирован событием, которое также отражается на диаграмме состояний. На Рисунке 14 приведен пример диаграммы состояний для банковского счета.

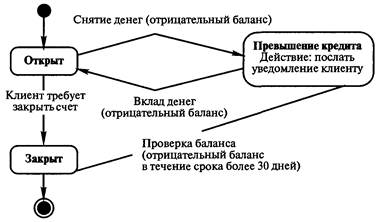


Рисунок 14

На данной диаграмме показаны возможные состояния счета, а также процесс перехода счета из одного состояния в другое. Например, если клиент требует закрыть открытый счет, последний переходит в состояние ***Закрыт***. Требование клиента называется *событием,* именно события вызывают переход из одного состояния в другое.

Когда клиент снимает деньги с открытого счета, счет может перейти в состояние ***Превышение кредита***. Это происходит, только если баланс по счету меньше нуля, что отражено условием отрицательный баланс.

На диаграмме имеются два специальных состояния — *начальное* и *конечное.*

Начальное состояние выделяется черной точкой: оно соответствует состоянию объекта в момент его создания.

Конечное состояние обозначается черной точкой в белом кружке: оно соответствует состоянию объекта непосредственно перед его уничтожением.

На диаграмме состояний может быть только одно начальное состояние. В то же время может быть столько конечных состояний, сколько вам нужно или их может не быть вообще.

Когда объект находится в каком-то конкретном состоянии, могут выполняться те или иные процессы. В нашем примере при превышении кредита клиенту посылается соответствующее сообщение. Процессы, происходящие, когда объект находится в определенном состоянии, называются *действиями.*

Поведение объекта, который должен отвечать на асинхронные сообщения, либо его текущее поведение зависит от прошлого, лучше всего специфицировать при помощи автоматов. *Автомат* – это описание последовательности состояний, через которые проходит объект на протяжении жизненного цикла, реагируя на события, а также описание реакции на эти события. Автоматы используются также для моделирования поведения систем в целом, особенно реактивных систем, которые должны отвечать на сигналы приходящие от объектов извне системы. Наиболее общее назначение автоматов – моделирование жизненного цикла объекта, в особенности если речь идет об экземплярах классов, вариантов использования и о системе в целом.

## Примеры реализации диаграмм в программном коде

### Диаграмма классов

Апплет на языке Java, выводящий строку «Здравствуй, мир!» в Web-браузере:

|  |  |
| --- | --- |
| **import java.awt.Graphics;**  **class HelloWorld extends java.applet.Applet{**  **public void paint (Graphics g) {**  **g.drawString(“Здравствуй, мир!”, 10, 10);**  **}**  **}** | Рисунок 15 |

### Диаграммы последовательности и коопераций

Пример метода на языке Java для регистрации студента на курсах, Рисунок 16.

**public void register() {**

**CourseCollection courses = getSchedule();**

**for(int i = 0; i < courses.size(); i++)**

**courses.item(i).add(this);**

**this.registered = true;**

**}**

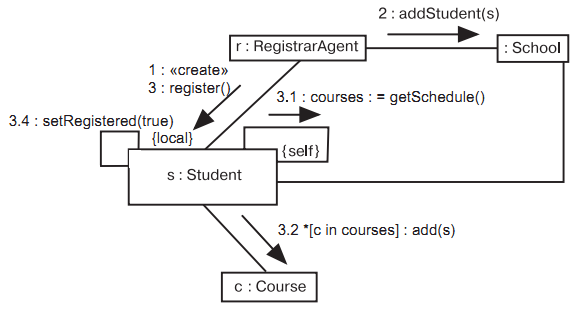


Рисунок 16

### Диаграммы деятельности

Пример метода на языке С++ для опеределения точки пересечения прямых, Рисунок 17.

**Point Line::intersection (Line : line) {**

**if (slope == line.slope) return Point(0,0);**

**int x = (line.delta – delta) /**

**(slope – line.slope);**

**int y = (slope \* x) + delta;**

return Point(x, y);

}

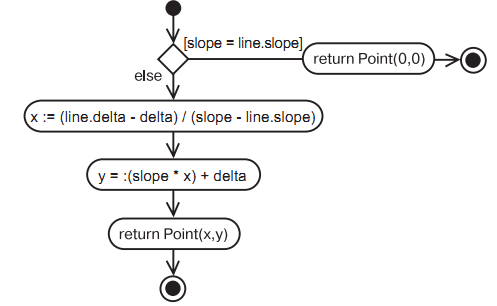


Рисунок 17

### Диаграммы состояний

Пример метода на языке Java для парсинга строки, Рисунок 18

**class MessageParser {**

**public**

**boolean put(char c) {**

**switch (state) {**

**case Waiting:**

**if (c == ‘<’) {**

**state = GettingToken;**

**token = new StringBuffer();**

**body = new StringBuffer();**

**}**

**break;**

**case GettingToken :**

**if (c == ‘>’)**

**state = GettingBody;**

**else**

**token.append(c);**

**break;**

**case GettingBody :**

**if (c == ‘;’)**

**state = Waiting;**

**else**

**body.append(c);**

**return true;**

**}**

**return false;**

**}**

**StringBuffer getToken() {**

**return token;**

**}**

**StringBuffer getBody() {**

**return body;**

**}**

**private**

**final static int Waiting = 0;**

**final static int GettingToken = 1;**

**final static int GettingBody = 2;**

**int state = Waiting;**

**StringBuffer token, body;**

**}**

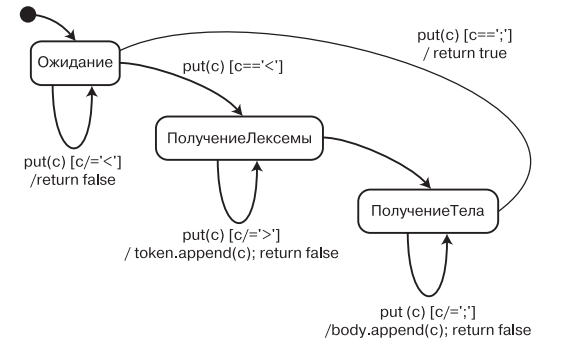


Рисунок 18

# Приложение Б. Инструменты для рисования UML-диаграмм

UML был разработан для определения, визуализации, проектирования и документирования программных систем. Инструменты для рисования UML-диаграмм позволяют по выполненным моделям классов генерировать программный код на различных языках программирования.

Ниже приводится список некоторых актуальных средств разработки UML-диаграмм.

1. MS Visio - <https://products.office.com/ru-RU/Visio?tab=tabs-1> (коммерческое ПО).
2. Sparx Enterprise Architect - [http://www.sparxsystems.com/](http://www.sparxsystems.com/products/ea/13/index.html#ea13.5) (Trial Edition for a 30 days).
3. Visual Paradigm – <https://www.visual-paradigm.com/> (No registration for 30-day FREE Trial).
4. Modelio SD 3.7.1 - <https://www.modeliosoft.com/en/download/download-products.html> (**trial** version for 10 days).
5. IBM Rational Rose – <https://www.ibm.com/developerworks/develop/> (не поддерживается IBM).
6. ArgoUML - <http://argouml.tigris.org/> (open source UML modeling tool)
7. StarUML 2 - <http://staruml.io/> (No time limit for evaluation, a license should be purchased for continued use).
8. Bouml\_7.5 - [bouml.fr](http://yandex.ru/clck/jsredir?bu=3bvxlf&from=yandex.ru%3Bsearch%2F%3Bweb%3B%3B&text=&etext=1770.ZJccHkYRT71yp9aYcpj3mmmMshyI6T4Q1eANT2mlg58.74969066b0edd48319386bee3cb3cf5f914c4260&uuid=&state=PEtFfuTeVD4jaxywoSUvtB2i7c0_vxGdjWT7Elao4JUWxZ-v0OIKpCInCIWESFN3UHbm4v3UYupETHesyjCjK_pRSpho5wC53vqJ5J-enwMx4rbV_3t91g,,&&cst=AiuY0DBWFJ5Hyx_fyvalFBAQgApwdMSZw43c8xbTAVQ8f5DmoKHg2ejDCPx_Sgfli5F5vGnzRxcZOv00xQXvALKxZqQ48Cb0f4f1h5L7AxvhmZ6z17ug1sRCAux8DDWAdJT9g6V0jtUg30LPG16rFGRS7ffL4zmCdLilzBEaVT-YA5qZqgQFPL2HG6EqdIqXemZDJWudtbpsokFtdSb8-bTHtHMjKKn7SQKJW1QcmqVLZGHldjYBRFqDO6cKin5kGb8AGk-df2vbPb26Cucrr30Dfo0ojhKvp1ASBYPxX5GI8m85n_EBf7V9nRMX9upv7T7puzJRyXi18vVpnEz9oG_E8X7U2Xif9xUOYwlBWfQgwI1PoRv-A78BgQQ5D4_AuHQAlcekygRO0iRpDjvjyG_8fhCqrvfgQagHmasd0So,&data=UlNrNmk5WktYejR0eWJFYk1LdmtxbTZGT1JpZXhhREZvd2REd3RiS1ZqN2M4dGZOV3ZIcVVGRFZwaFFQem9oZlZ0eFNyOGVFYzd3T21nYkhldmNvSnJQVF9sNC1VMVZi&sign=f5ca6317c618d3201ad08497dc90e473&keyno=0&b64e=2&ref=orjY4mGPRjk5boDnW0uvlrrd71vZw9kpjly_ySFdX80,&l10n=ru&cts=1524901497186&mc=5.226456566827983) (Since the release 7.0 BOUML is again a free software).

Приведем пример UML-диаграммы классов из папки ***Samples***, полученной в ***StarUML 2***, Рисунок 19.

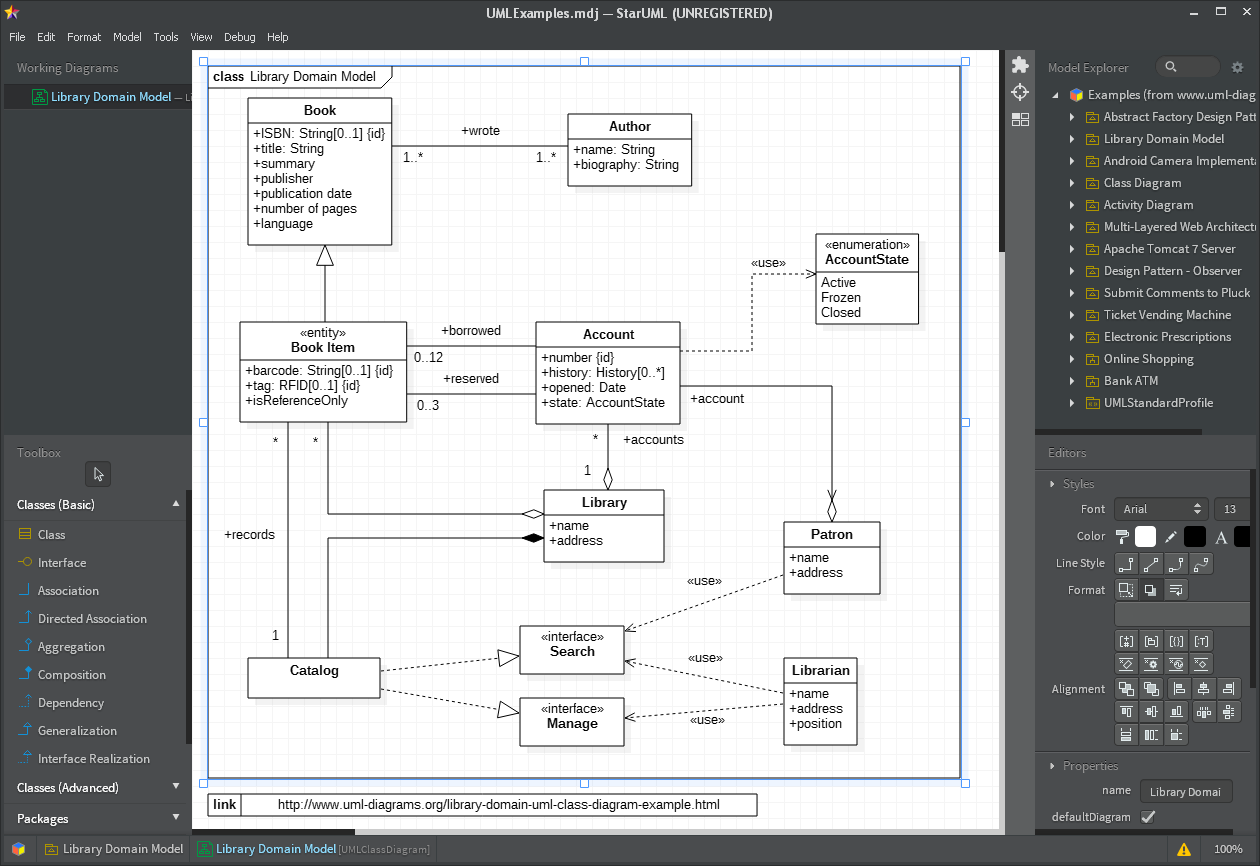


Рисунок 19

# Приложение В. Лист выдачи задания на курсовую работу

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Уфимский государственный нефтяной технический университет»

Кафедра «Цифровые технологии и моделирование»

ЗАДАНИЕ на курсовую работу

по дисциплине «Информатика и программирование»

студенту \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия Имя Отчество)

Тема работы: «Разработка приложения в среде Delphi», вариант \_\_\_\_

1. Назначение разработки: программа будет использоваться в учебном процессе для решения инженерных задач обработки массивов, путем ее включения в программы пользователей.
2. Технические требования к программному модулю:
   1. Приложение должно иметь интерфейс пользователя и модули с програмным кодом, соответствующие варианту задания.
   2. В программе должен быть предусмотрен необходимый комплект процедур и функций с параметрами для решения основных и вспомогательных подзадач.
   3. Все массивы должны быть объявлены как динамические.
   4. Предусмотреть необходимые комментарии к программному коду.
   5. Требования к составу и оформлению программной документации – типовые, предусмотренные методическими указаниями.
3. Указания по программированию:
   1. Программа должна иметь модульную структуру с подцеплением всех модулей к модулю главной формы.
   2. Предусмотреть защиту от неверно вводимых значений исходных данных.
4. Технические условия:
   1. Язык программирования: Object Pascal, инструментальная среда Delphi.
   2. При необходимости тестирования функциональности программного модуля, либо отдельных процедур, должен быть разработан соответствующий проект (программа).
   3. При тестировании приложения просчитать случайные, а также крайние значения параметров (сделать скрин-шоты результатов),
5. Стадии и этапы разработки:
   1. Выдача задания на курсовую работу \_\_\_\_\_неделя
   2. Разработка алгоритмов \_\_\_\_\_\_неделя
   3. Разработка программы \_\_\_\_\_\_неделя
   4. Отладка программы и оформление пояснительной записки \_\_\_\_\_неделя
   5. Защита курсовой работы \_\_\_\_неделя

Руководитель проекта, преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Группа) (подпись) (ФИО)

Дата выдачи “\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

# Приложение Г. Титульный лист курсовой работы

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Уфимский государственный нефтяной технический университет»

**Кафедра «Цифровые технологии и моделирование»**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| График выполнения КР | | | | | | | |
| **Недели** | \_\_ | \_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |
| 100 % |  |  |  |  |  |  |  |
| 80 % |  |  |  |  |  |  |  |
| 60 % |  |  |  |  |  |  |  |
| 40 % |  |  |  |  |  |  |  |
| 20 % |  |  |  |  |  |  |  |
| Оценка | ритмичности: \_\_\_\_\_\_\_\_\_ | | | | | | |
| оформления: \_\_\_\_\_\_\_\_\_ | | | | | | |
| защиты: \_\_\_\_\_\_\_\_\_\_\_ | | | | | | |
| **Общая оценка** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | |

**Структурное и объектно-ориентированное моделирование   
процессов и систем**

Пояснительная записка к курсовой работе по дисциплине

**Программная инженерия**

Вариант Код\_\_\_\_\_

Выполнил(а) студент(ка) группы \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И.О.

(Дата, подпись)

Принял преподаватель каф. ЦТиМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И.О.

(Дата, подпись)

Уфа – 2018

# Список литературы

1. Боггс У., Боггс М. UML и Rational Rose. - М.: Лори, 2001. - 582 с.
2. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2006. – 496 с.
3. Орлов С.А. Программная инженерия. Учебник для вузов. 5-е издание обновленное и дополненное. Стандарт третьего поколения. – СПб.: 2016. – 640 с.
4. Дмитрий К. Введение в программную инженерию. https://www.intuit.ru/studies/courses/497/353/info.
5. [Ефименко](https://www.intuit.ru/intuituser/demands/author/504041) В. Методы и средства инженерии программного обеспечения. https://www.intuit.ru/studies/professional\_skill\_improvements/17207/info.
6. Радченко Г.И. Программная инженерия. http://glebradchenko.susu.ru/courses/bachelor/engineering/2016.
7. Основы программной инженерии [Электронный ресурс] : Учебно-методическое пособие (для выполнения лабораторных работ) /