МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Московский физико-технический институт
(государственный университет)

Кафедра алгоритмов и технологий программирования

КОМАНДНЫЙ ПРОЕКТ ПО ОБЪЕКТНО-ОРИЕНТИРОВАННОМУ ПРОЕКТИРОВАНИЮ

Методическое пособие

А.С. Хританков

МОСКВА
МФТИ
2012

УДК 004.41+004.02+372.8
ББК 32.973.23-018

Рецензент:
к.т.н Рейер И.А.

Командный проект по объектно-ориентированному проектированию: Методическое пособие / А.С. Хританков − М.: МФТИ, 2012. − 35 с.

В данном пособии изложены рекомендации по выполнению командных самостоятельных заданий по объектно-ориентированному проектированию. Приведены рекомендуемые структуры отчетов по заданиям, виды ошибок и критерии выставления оценок.

Для студентов высших учебных заведений физико-математических и технических специальностей.

УДК 004.41+004.02+372.8
ББК 32.973.23-018

Содержание

Введение 5
Критерии выставления оценок 8
Структура отчета по первому заданию 11
Структура отчета по второму заданию 21

ПРЕДИСЛОВИЕ

Основная задача данного пособия – изложить в доступной форме подробные методические рекомендации по выполнению командных заданий по объектно-ориентированному проектированию и восполнить недостаток внимания, уделяемого данному вопросу в методической литературе.

Пособие состоит из введения, трех разделов и списка литературы. Во введении дается краткая характеристика курса, приведены темы курса и пояснения по курсовому командному проекту. Проект выполняется студентами в группах самостоятельно. При выполнении проекта в первом задании студенты по исходному описанию системы разрабатывают модели взаимодействия системы с пользователями и другими системами. Во втором задании студенты уточняют и дополняют модель для выбранной программной платформы.

В первом разделе приведены критерии выставления оценок по курсовому проекту. Во втором и третьем разделах пособия даны подробные пояснения по решению первого и второго задания по курсовому проекту соответственно.

Приведенные в данном пособии диаграммы UML выполнены с помощью программы MagicDraw, предоставленной компанией No Magic Inc. в рамках академической программы.

Пособие предназначено для студентов, обучающихся по специальности 010900 «Прикладные математика и физика».

ВВЕДЕНИЕ

1. Описание курса

Курс «Объектно-ориентированные технологии проектирования и язык UML» посвящен изучению современных подходов к разработке сложных приложений и программных комплексов. В рамках курса студенты изучают унифицированный язык моделирования и на его основе методы объектно-ориентированного анализа и проектирования, знакомятся с методами структурного проектирования. Программа курса в достаточной степени отвлечена от конкретных технологий, чтобы дать адекватное представление о дисциплине объектно-ориентированного проектирования ПО. В результате обучения студенты знакомы с методами проектирования, способны самостоятельно разработать согласованную модель программной системы, удовлетворяющей функциональным требованиям, и обосновать принятые проектные решения, что подтверждается самостоятельным выполнением курсовых заданий.

Изучение курса требует предварительных знаний в следующих областях: информатика, объектно-ориентированное программирование, технология разработки ПО. Желательно владение техническим английским языком, так как часть материалов по курсу представлена на английском языке.

Цель курса состоит в ознакомлении студентов с процессом разработки программных систем с применением объектно-ориентированных технологий и технологий моделирования. Основные задачи курса: дать представление о существующих методологиях проектирования программного обеспечения и выработать у студентов практические навыки по их применению. Уклон делается на разработку прикладного программного обеспечения.

2. Темы курса

Введение в инженерию программного обеспечения. Различия между программным и аппаратным обеспечением. Показатели качества ПО. Виды ПО. Цели и задачи проектирования. Отличия проектирования от исследования. Основные определения в инженерии ПО. Процесс, проект, рабочий продукт. Основные процессы разработки. Предписывающие и описывающие процессы. Каскадная, инкрементная, итеративная модели жизненного цикла. Организация разработки ПО.

Основы моделирования при помощи UML. Понятие о моделировании и проектировании. Структурное моделирование. История создания

UML. Структура метамодели UML. Основные понятия: элемент, классификатор, отношение, пространство имен. Представления модели, виды диаграмм.

Статическое представление модели. Диаграммы классов. Понятия класса, интерфейса, типа данных. Виды отношений: ассоциация, зависимость, абстракция, реализация и другие. Ограничения. Экземпляры классов. Профили. Расширение модели. Варианты использования (прецеденты). Описание требований при помощи прецедентов. Пакеты. Управление моделью. Структурированный классификатор. Композит и часть. Диаграммы внутренней структуры.

Динамическое представление модели. Поведение. Основные определения. Представление взаимодействия. Диаграммы взаимодействия и коммуникации. Основные понятия: роль, спецификация выполнения, сообщение. Кооперация. Описание сценариев вариантов использования. Представление деятельности. Общие сведения о сетях Петри. Виды действий, разделы. Контекст выполнения. Потоки управления и данных (объектные). Представление процессов на диаграммах деятельности. Представление конечных автоматов. Диаграммы схем состояний. Состояние, переход, псевдосостояния, составные состояния. Семантика конечных автоматов в UML. Обработка событий, переход по завершении. Моделирование жизненного цикла классификатора с помощью конечных автоматов.

Введение в архитектуру ПО. Архитектура и структура программной системы. Факторы, влияющие на архитектуру. Заинтересованные лица. Роль архитектуры. Стандартные стили архитектур: вызов-возврат, каналы-фильтры, многоуровневая, клиент-сервер, сервис-ориентированная. Совместное применение стилей.

Методы структурного проектирования. Основные понятия. Модуль. Процесс структурного проектирования. Виды методов: сверху-вниз (низходящие), снизу-вверх (восходящие), итеративные. Модульность. Принципы разделения системы на модули. Метрики качества модульной структуры. Метод постепенного уточнения, структурные диаграммы (STD). Методика структурного анализа/проектирования (SSA/SD). Элементарные транзакции. Диаграммы потоков данных (DFD). Метод структурного программирования Джексона (JSP). Разбор примеров применения.

Объектно-ориентированное проектирование. Абстрактные типы данных. Принципы ООП. Введение в объектно-ориентированный анализ. Выделение классов. Метод Аббота, карточки Класс-Контракт-Коллеги (CRC), диаграммы устойчивости. Эвристики построения моделей классов. Характеристики качества объектно-ориентированной структуры программы. Паттерны проектирования. Структурные, создания и паттерны пове-

дения (GoF). Примеры паттернов. Строитель. Посетитель. Шаблон метода. Фасад. Мост. Эвристики GRASP.

Введение в архитектуру ПО. Архитектура и структура программной системы. Факторы, влияющие на архитектуру. Заинтересованные лица. Роль архитектуры. Стандартные стили архитектур: вызов-возврат, каналы-фильтры, многоуровневая, клиент-сервер, сервис-ориентированная. Совместное применение стилей.

Документирование архитектуры и структуры программ. Роль документирования. Понятие о перспективе и представлении. Рекомендации по составлению и структура документа с описание архитектуры и дизайна.

Применение моделей в разработке программ. Место моделей в процессах жизненного цикла проектов. Обзор методики ICONIX, основные этапы, применяемые методы и эвристики. Применение моделей в гибкой разработке.

Проблемы современной практики проектирования. Модельно-ориентированный подход к разработке (MDD/MDA). Платформонезависимая модель и платформозависимая модель (PIM/PSM). Сервис-ориентированная архитектура. Потоки работ (workflow). Использование каркасов приложений (framework). Проблемы проектирования распределенных объектных систем. Понятие о транзакциях.

3. Командные курсовые задания для самостоятельной работы

Существенной частью курса являются практические задания, выполняемые в группах самостоятельно вне аудиторных занятий. Далее такие задания называются курсовыми проектами.

Задания выполняются в два этапа. Первый этап направлен на разработку модели взаимодействия системы с пользователями и другими системами. На втором этапе модель уточняется и дополняется решениями по реализации системы на выбранной программной платформе.

В данном пособии изложены рекомендации по выполнению курсовых проектов. Приведены рекомендуемые структуры отчетов по первому и второму этапу, виды ошибок и критерии выставления оценок.

В структуре отчетов примеры заполнения разделов приведены курсивом, как следующем предложении.

Это пример заполнения раздела в структуре отчета.

КРИТЕРИИ ВЫСТАВЛЕНИЯ ОЦЕНОК

1. Замечания и комментарии по заданиям

Логические ошибки

Логические ошибки и упущения, неверное применение методов:

-) отсутствует класс Билет;

-) в классе отсутствует вызываемая операция;

--) описанная деятельность не может быть выполнена ни одного раза;

--) выделены не все варианты использования.

Количество "-" указывает на серьезность ошибки.

Формальные ошибки моделирования и представления

Недочеты по форме представления, ошибки в синтаксисе UML (которые не приводят к существенным логическим ошибкам)

*) не указано название полюса ассоциации;

*) класс не определяет операции, реализующие интерфейс;

*) неверно указана кратность полюса;

*) в CRC не указываются операции и атрибуты, а только обязанности классов;

**) использована неверная диаграмма.

Количество «*» указывает на серьезность недочета.

Удачные моменты и решения

+) правильно выделены классы;

+) у двух сходных классов выделен общий предок;

+) к месту применено описание процесса в виде деятельности;

++) даны подробные пояснения и обоснования по проекту.

2. Рекомендации по выставлению оценок по курсовым заданиям

Первое и второе задания рекомендуется оценивать по десятибалльной шкале.

Оценка отлично (десять) ставится в исключительных случаях, когда студенты продемонстрировали владение предметом, выходящие за рамки курса.

По заданиям не предусматривается возможность повторного выполнения и защиты, поэтому задание считается сданным в случае получения

отчета с решением преподавателем. Если отчет не был получен, оценка за задание полагается равной нулю.

Что такое отлично выполненное задание?

При выполнении указанных требований задание заслуживает оценки отлично (девять):

- Отчет по заданию оформлен согласно шаблону.
- Методы проектирования применены правильно.
- Принятые при выполнении задания решения обоснованы.
- Ошибок в UML нет.
- Решение согласуется с постановкой задачи.
- Модель непротиворечива и полна.
- Проведена презентация, изложены основные решения.
- В команде понятно, кто что делал. Участники четко ответили на все вопросы на защите задания.
- Задание выполнено командой самостоятельно.
- Задание сдано в срок.

Снижение оценки за недоработки и неудачные решения

Оценка за задание может быть снижена в следующих случаях:

- Не выдержаны сроки выполнения задания.
- При выполнении задания команда не продемонстрировала полноту владения методами, ограничившись частными случаями.
- Вообще говоря, решение задания не является полностью оригинальным.
- Принятые проектные решения неудачны или не подходят для решения задания.

Что такое хорошо выполненное задание?

При выполнении указанных требований задание заслуживает оценки хорошо (шесть):

- Отчет по заданию в целом оформлен согласно шаблону.
- Имеются следы применения методов проектирования, методы применены с ошибками.
- Некоторые решения обоснованы, обоснования многих решений недостаточные.
- На диаграммах имеются ошибки в UML, не влияющие на понимание модели.
- Решение в целом соответствует исходной постановке задачи, могут быть отличия, не влияющие в целом на задание.
- Модель непротиворечива, но не является полной - отсутствует описание некоторых элементов.

- Презентация проведена, но из нее сложно понять, что было сделано командой.
- В команде понятно, кто что делал. Участники смогли ответить на поставленные вопросы.
- Задание выполнено командой самостоятельно.
- Задание сдано в срок или с небольшим опозданием.

Повышение оценки за удачные решения

Оценка за задание может быть повышена в следующих случаях:

- В ходе выполнения задания команды применила методы, выходящие за рамки курса.
- Команда провела дополнительные самостоятельные исследования, результаты которого использованы при выполнении задания.
- При выполнении задания команда продемонстрировала более глубокие знания по предмету, чем требуется по программе курса.
- Приняты удачные проектные решения, соответствующие современной практике проектирования.

Увеличение объема работы не должно рассматриваться как основание для повышения оценки.

Что такое плохо выполненное задание?

При выполнении указанных требований задание заслуживает оценки удовлетворительно (два):

- Отчет по заданию оформлен не по шаблону, некоторые разделы верхнего уровня отсутствуют.
- Методы проектирования применены неправильно или отсутствует упоминание методов.
- Принятые при выполнении задания решения не обоснованы или противоречивы, не имеют смысла.
- Имеются существенные ошибки в использовании UML.
- Решение не соответствует исходной постановке задачи, имеются существенные расхождения.
- Построенная модель не согласована, составлена из обрывков или несвязанных частей.
- Презентация задания не проведена, либо команда не готова к презентации.
- В команде не понятно, кто что делал. Участники не смогли ответить на вопросы по заданию, либо отвечали с значительными ошибками, давали противоречивые ответы.
- Задание сдано существенно позже назначенного срока.

СТРУКТУРА ОТЧЕТА ПО ПЕРВОМУ ЗАДАНИЮ

1. Постановка задачи

1.1. Состав первого задания

Своими словами изложить, в чем состоит первое задание. Каждая команда делает самостоятельно.

1.2. Описание задачи

Привести текст с постановкой задачи.

Покупателям нужна возможность забронировать билет в театр из дома. При этом будет полезно сначала посмотреть афишу театра. Афиша включает перечень спектаклей, по каждому из которых имеется расписание представлений. Необходимо предусмотреть также возможность бронирования покупателями нескольких билетов на одно представление и бронирования абонемента на несколько представлений. Билет на представление бронируется на конкретное место в зале с указанием ряда и кресла. На одно место не может быть продано более одного билета.

2. Описание предметной области

2.1. Идентификация классов

Провести идентификацию классов с помощью одного из известных методов. Например, Аббота или именных групп. Дать пояснения по шагам применения метода.

Для идентификации классов и построения первого варианта модели предметной области воспользуемся методом Аббота. Потенциальные классы приведены в таблице 1.
Условные обозначения критериев проверки классов:
- *С : класс сохраняет информацию;*
- *И : предполагается наличие интерфейса для изменения хранимой информации;*
- *А : для хранения информации используется несколько атрибутов;*
- *О : класс реализует несколько действий;*

- *У : атрибуты и операции класса применимы ко всем экземплярам;*
- *Т : наличие класса в модели является существенным требованием.*

Таблица 1. Потенциальные классы и проверка критериев

Потенциальный класс	***Критерии***	***Хранимая информация***	***Выделенные действия***
Представление	*СУ*	*Дата*	*Бронировать билет*
Спектакль	*СУТ*	*Представление*	
Билет	*САУТ*	*Место* *Представление*	
Абонемент	*СУТ*	*Билеты*	
Место	*САУ*	*Ряд* *Кресло*	
Афиша театра	*СУ*	*Спектакли*	*Посмотреть*
Покупатель			

Дать пояснения по составлению таблицы:

- какие существительные объединены, как синонимы;
- какие атрибуты предполагается использовать для хранения информации;
- как были получены действия.

Описать действия по анализу таблицы и принятию решений по классам.

Потенциальный класс Спектакль хранит информацию о расписании представлений. Будем предполагать, что в театре только один зал, тогда для хранения расписания спектаклей достаточно знать даты, по которым будет дан спектакль.
Класс Покупатель был исключен из модели. Действие Посмотреть класса Афиша театра было исключено из дальнейшего рассмотрения. так как реализуется чтением хранимой классом информации.
В дальнейшем класс Покупатель может быть представлен как актор, использующий систему.

2.2. Предварительная логическая модель бронирования билетов

Построить диаграмму классов, указать, каким образом были получены атрибуты, ассоциации и операций из хранимой информации и действий. При разработке модели названия классам следует давать на английском языке, при этом должно быть понятно соответствие между классами и потенциальными классами.

Также нужно дать пояснения по выделенным ассоциациям и кратностям полюсов. При оформлении рисунков следует придерживаться одного выбранного масштаба, например, 75%.

Между Афишей театра Repertoire добавлена ассоциация с полюсом plays у спектакля Play. В одной афише может быть несколько спектаклей, поэтому соответствующая кратность. Указание агрегации обосновано тем, что хранимая информация Афиши театра состоит из несколько Спектаклей.
Диаграмма классов модели приведена на рис. 2.

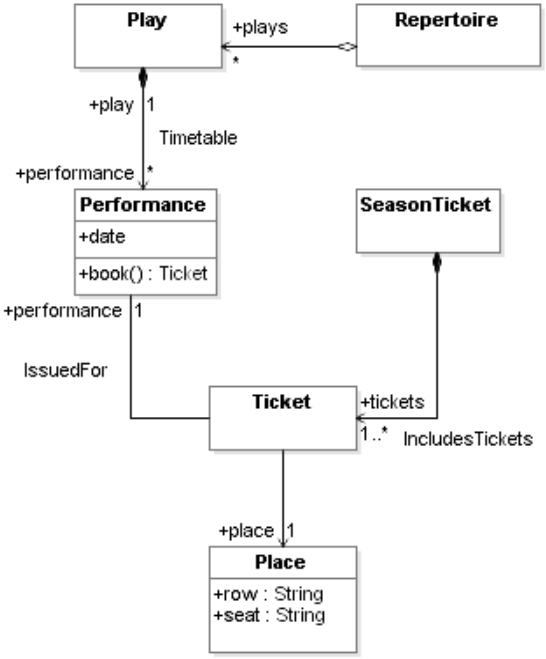

Рис. 1. Предварительная диаграмма классов бронирования билетов

2.3. Обновленная логическая модель бронирования билетов

Данный раздел заполняется после анализа взаимодействий. Далее привести обновленную диаграмму классов. Рекомендуется сохранить предыдущую версию проекта (или в отдельном пакете), так как обосновать обновленную диаграмму в разделе 2.2 будет нельзя. В разделе 2.2 следует использовать старый вариант.

На рис. 2 приведена диаграмма классов после рассмотрения вариантов использования и анализа взаимодействий.

Далее рассказать, каким образом были совмещены изменения по отдельным взаимодействиям, как были устранены противоречия (если возникли). Следует приводить только окончательный вариант диаграммы. Промежуточные нужно перенести в приложение.

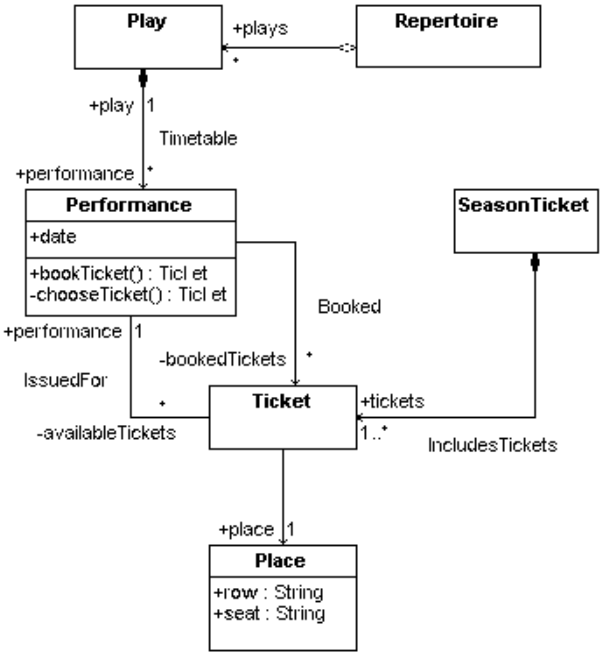

Рис. 2. Обновленная диаграмма классов бронирования билетов

3. Описание требований к системе

3.1. Модель вариантов использования

Здесь нужно перечислить акторов, варианты использования и построить диаграмму вариантов использования. Дать обоснование по выделенным акторам, их целям. Указать, какого уровня будут варианты использования. Обратите внимание на использование терминов и классов предметной области.

В описании системы упоминается Покупатель. Покупателю доступна возможность бронирования билетов на представления и бронирования абонементов, включающих билеты на несколько представлений. Кроме того, покупатель может посмотреть афишу театра.
Вариант использования BookTicket объединяет сценарии бронирования билета покупателем. Данные сценарии отличаются от бронирования абонемента, так как покупатель сначала выбирает Представление, а не выбирает из имеющихся абонементов.
Просмотр афиши выделен в отдельный вариант использования потому, что информация о Спектаклях театра важна для Покупателя сама по себе. Тем не менее, при бронировании билета, Покупатель может посмотреть Афишу театра, поэтому указано отношение включения.

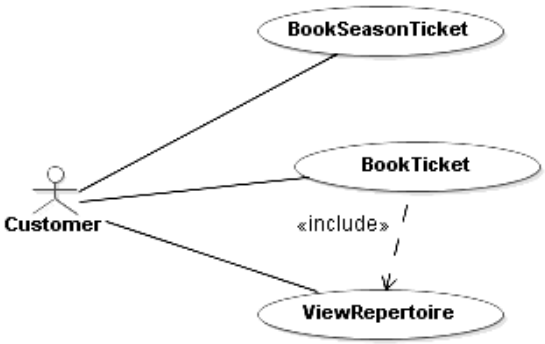

Рис. 3. Диаграмма вариантов использования.

3.2. Вариант использования BookTicket

В данном разделе нужно подробно описать вариант использования в соответствии со структурой

- Акторы.
- Цель.

- Предусловия.
- Постусловия.
- Основной сценарий.
- Альтернативные сценарии.
- Точки расширения.

Акторы: *Customer (Покупатель)*
Цель: *Покупатель хочет забронировать билет на определенное представление из репертуара театра.*
Предусловия: *Афиша театра содержит хотя бы один спектакль, по которому есть представления.*
Постусловия: *Покупатель забронировал билет.*
Основной сценарий:

1. *Покупатель запрашивает афишу театра. Система показывает Покупателю афишу театра.*
2. *Покупатель выбирает спектакль в афише театра и запрашивает список дат представлений (расписание представлений). Система возвращает список дат представлений для выбранного спектакля.*
3. *Покупатель выбирает дату представления и бронирует билет. Система выбирает место и возвращает покупателю билет на указанную дату с выбранным местом.*

Альтернативные сценарии:
Если на шаге 1. покупатель решает посмотреть несколько спектаклей, то выполняется вариант использования ViewRepertoire.

3.3. Вариант использования BookSeasonTicket

Аналогично предыдущему пункту.

3.4. Вариант использования ViewRepertoire

Аналогично предыдущему пункту.

4. Моделирование взаимодействий и поведения

В данном разделе нужно провести анализ взаимодействий, описанных в вариантах использования. Это означает, что для каждого варианта использования нужно проработать, какие классы будут участвовать в его реализации, и какие обязанности они будут иметь. Что делать: выделить взаимодействия, для каждого используя карточки CRC выделить и назначить обязанности, затем построить диаграммы последовательности. Не забыть реализовать обязанности в виде операций и атрибутов, показать их на диаграмме классов.

На данном этапе в модель нужно добавить кооперации, реализующие варианты использования. В кооперациях будут участвовать акторы. Раздел заполняется после анализа взаимодействий в подразделах.

Варианты использования BookTicket, BookSeasonTicket и ViewRepertoire могут быть реализованы кооперацией Booking.
Кооперация Booking содержит покупателя Customer, афишу Repertoire, выбранный покупателем спектакль Play и выбранное представление Performance.

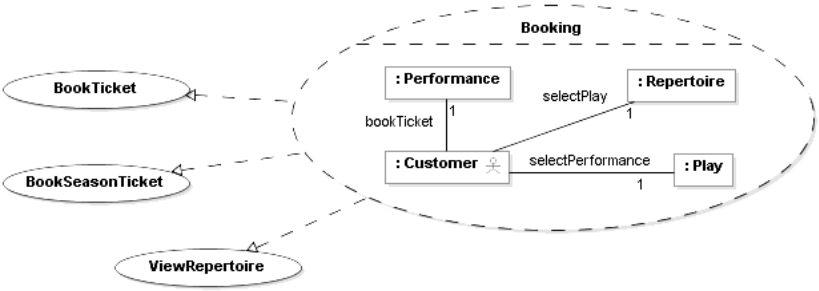

Рис. 4. Реализация вариантов использования с помощью кооперации Booking.

4.1. Реализация варианта использования BookTicket

Провести анализ основного и альтернативных сценариев с помощью карточек CRC или другого метода, например, диаграмм понимания (robustness). Оформить в таблицу.

Обязанности формулировать по типам:

- хранить (и управлять), знать, предоставлять (но не хранить) информацию, знать коллег;
- использовать (для получения данных), выполнять действия, вызывать действия в других объектах;
- предоставлять интерфейс.

Следует использовать имеющуюся модель предметной области в качестве заготовки. Ее можно дополнять. Обязанности нужно формулировать, используя имеющиеся в ней классы. Если для назначения обязанности класса не хватает (нет подходящего), то его нужно добавить и дать обоснование.

Результаты анализа основного сценария с помощью карточек CRC приведены в таблице 2.

Таблица 2. Обязанности и коллеги участников реализации варианта использования BookTicket.

Объект класса	***Контракт***	***Коллеги***
Customer	*Использует афишу* *Использует спектакль* *Вызывает представление*	*Repertoire* *Play* *Performance*
Repertoire	*Хранит список спектаклей*	*Play*
Play	*Хранит список представлений*	*Performance*
Performance	*Хранит дату представления* *Знает доступные для бронирования билеты* *Хранит забронированные билеты* *Выбирает билет из доступных* *Бронирует билет*	*Ticket*
Ticket	*Хранит место* *Знает представление*	*Performance* *Place*

Затем дать пояснения, каким образом обязанности уже отражены в модели предметной области. Указать, какие не отражены. Их разложить на атрибуты, ассоциации и операции и добавить в модель.

Обязанность класса Performance по хранению доступных и забронированных билетов в модели реализуется двумя ассоциациями: IssuedFor и Booked. Обязанность бронирования билетов реализуется операцией bookTicket(), которая выбирает билет из доступных частной операцией chooseTicket() и переносит его из IssuedFor в Booked, сохраняя состояние бронирования билета Ticket.
Остальные обязанности уже отражены в модели предметной области. Изменения в модели показаны на рис. 5.

Следует приводить только фрагмент диаграммы классов, на котором показаны описанные в тексте изменения.

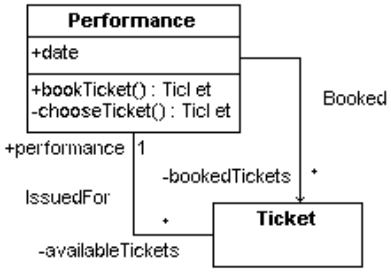

Рис. 5. Изменения в модели бронирования билетов после реализации BookTicket

4.2. Реализация варианта использования BookSeasonTicket

Аналогично предыдущему пункту.

4.3. Реализация варианта использования ViewRepertoire

Аналогично предыдущему пункту.

4.4. Описание собственного поведения объектов класса А

В некоторых случаях объекты предметной области обладают собственным поведением. Например, классов лифт, переезд. Для описания их поведения следует составить схему состояний. На данном этапе диаграмма служит для пояснения и прояснения поведения. Важно показать состояния, события и условия переходов. Здесь нужно привести схему состояний, дать пояснения в тексте: обосновать, почему выделено каждое состояния, переходы между состояниями.

4.5. Описание алгоритма или процесса Б

В других случаях для описания предметной области может быть важно показать алгоритм выполнения какого-либо действия или процесса. Для этого можно использовать диаграммы деятельности. На диаграмме обозначить основные шаги процесса, последовательность действий, зависимости по управлению. Для назначения действий участникам (акторам, классам) рекомендуется использовать разделы (плавательные дорожки).

Здесь привести диаграмму деятельности, дать пояснения по действиям и разделам.

5. Приложение 1. Артефакты проектирования

В приложении нужно привести фотографии / копии артефактов, использованных при разработке, которые подтверждают применение методов проектирования.

Хорошим примером являются карточки CRC, обсуждения на бумаге проектных решений.

СТРУКТУРА ОТЧЕТА ПО ВТОРОМУ ЗАДАНИЮ

1. Постановка задачи

1.1. Состав второго задания

Своими словами изложить, в чем состоит второе задание. Каждая команда делает самостоятельно.

1.2. Описание задачи

Привести текст с постановкой задачи.

Покупателям нужна возможность забронировать билет в театр из дома. При этом будет полезно сначала посмотреть афишу театра. Афиша включает перечень спектаклей, по каждому из которых имеется расписание представлений. Необходимо предусмотреть также возможность бронирования покупателями нескольких билетов на одно представление и бронирования абонемента на несколько представлений. Билет на представление бронируется на конкретное место в зале с указанием ряда и кресла. На одно место не может быть продано более одного билета. Система доступна для использования через Интернет. Интерфейс пользователя отображается средствами веб-браузера. Если покупатель использует мобильный браузер, то ему отображается упрощенная версия пользовательского интерфейса.

2. Описание предметной области

2.1. Логическая модель бронирования билетов

Данный раздел приводится из задания 1. Привести последнюю версию диаграммы классов.

На рис. 2 приведена диаграмма классов после рассмотрения вариантов использования и анализа взаимодействий.

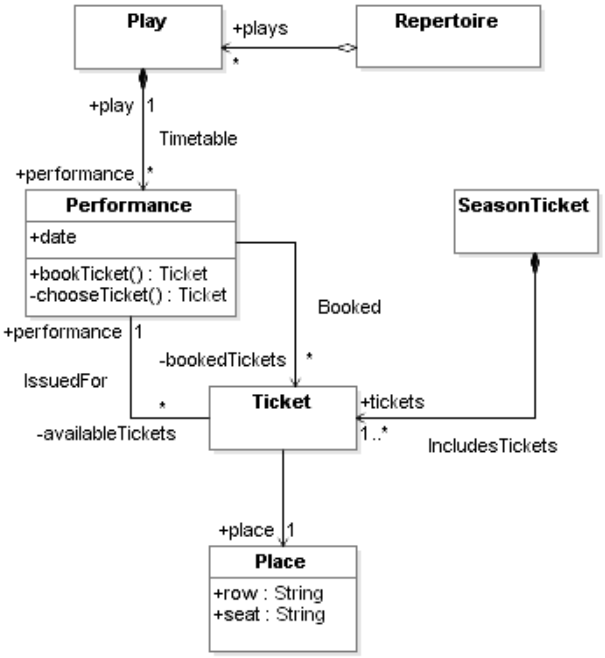

Рис. 6. Обновленная диаграмма классов бронирования билетов

2.2. Описания и обязанности классов

Привести таблицу с перечнем классов и обязанностями, коллегами. Обязанностей и коллеги приводятся из первого задания, агрегированно по всем взаимодействиям.

Таблица 3. Обязанности и коллеги классов предметной области бронирования билетов.

Класс	***Контракт***	***Коллеги***
Customer	*Использует афишу* *Использует спектакль* *Вызывает представление*	*Repertoire* *Play* *Performance*
Repertoire	*Хранит список спектаклей*	*Play*
Play	*Хранит список представлений*	*Performance*

Performance	*Хранит дату представления* *Знает доступные для бронирования билеты* *Хранит забронированные билеты* *Выбирает билет из доступных* *Бронирует билет*	*Ticket*
Ticket	*Хранит место* *Знает представление*	*Performance* *Place*

3. Описание функциональных требований к системе

3.1. Модель вариантов использования

Раздел повторяет аналогичный раздел из задания 1. См. пояснения в описании первого задания.

В описании системы упоминается Покупатель. Покупателю доступна возможность бронирования билетов на представления и бронирования абонементов, включающих билеты на несколько представлений. Кроме того, покупатель может посмотреть афишу театра.
Вариант использования BookTicket объединяет сценарии бронирования билета покупателем. Данные сценарии отличаются от бронирования абонемента, так как покупатель сначала выбирает Представление, а не выбирает из имеющихся абонементов.
Просмотр афиши выделен в отдельный вариант использования потому, что информация о Спектаклях театра важна для Покупателя сама по себе. Тем не менее, при бронировании билета, Покупатель может посмотреть Афишу театра, поэтому указано отношение включения.

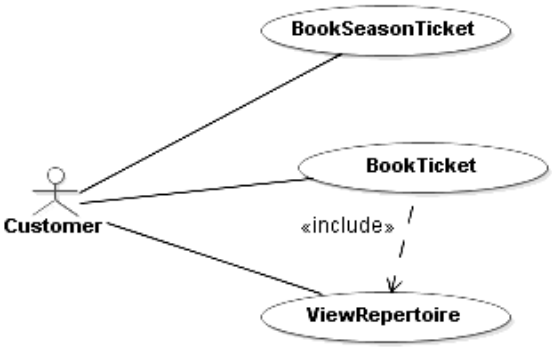

Рис. 7. Диаграмма вариантов использования.

3.2. Вариант использования BookTicket

В данном разделе нужно подробно описать вариант использования в соответствии со структурой

- Акторы.
- Цель.
- Предусловия.
- Постусловия.
- Основной сценарий.
- Альтернативные сценарии.
- Точки расширения.

Акторы: *Customer (Покупатель)*
Цель: *Покупатель хочет забронировать билет на определенное представление из репертуара театра.*
Предусловия: *Афиша театра содержит хотя бы один спектакль, по которому есть представления.*
Постусловия: *Покупатель забронировал билет.*
Основной сценарий:

1. *Покупатель запрашивает афишу театра. Система показывает Покупателю афишу театра.*
2. *Покупатель выбирает спектакль в афише театра и запрашивает список дат представлений (расписание представлений). Система возвращает список дат представлений для выбранного спектакля.*
3. *Покупатель выбирает дату представления и бронирует билет. Система выбирает место и возвращает покупателю билет на указанную дату с выбранным местом.*

Альтернативные сценарии:

Если на шаге 1. покупатель решает посмотреть несколько спектаклей, то выполняется вариант использования ViewRepertoire.

3.3. Вариант использования BookSeasonTicket

Аналогично предыдущему пункту.

3.4. Вариант использования ViewRepertoire

Аналогично предыдущему пункту.

4. Описание архитектуры системы

4.1. Описание трехуровневой архитектуры

Привести название и описание выбранного архитектурного стиля. Перечень стилей и их описание можно взять у преподавателя.

Система состоит из трех основных компонентов: клиент, предоставляющий пользовательский интерфейс; сервер, на котором располагается бизнес-логика системы; и база данных, которая хранит данные системы в структурированном виде.
Сервер принимает запросы от клиента через HTTP порт и направляет запросы на обработку сервлетам, входящим в состав сервера. Сервлет может быть контроллером, обрабатывающим запросы, либо страницей сервера, которая формирует HTML страницу для отправки клиенту.
Сервер предоставляет сервлетам порт для доступа к базе данных с операциями CRUD и порт для управления отображением страниц сервера с операциями создания и отображения страниц.
База данных сохраняет объекты доменной области, обладающие уникальным идентификатором. Для описания структуры сохраняемой информации используется схема, формируемая по логической модели приложения.

4.2. Обоснование выбора архитектурного стиля

Привести подробное обоснование, почему был выбран данный архитектурный стиль. Обоснование вида «так было сказано» не является достаточным. Нужно привести рассуждения, основанные на предполагаемом использовании системы, применяемых стилях в схожих системах.

Среди рассмотренных архитектурных стилей:

1. *трехуровневая архитектура*
2. *монолитное приложение*
3. *встраиваемая система*
4. *…*

Для разрабатываемой системы бронирования билетов лучше подходит стиль трехуровневой архитектуры, так как в нем возможно удаленное взаимодействие пользователей с системой через Интернет. Аналогичную архитектуру имеют другие системы бронирования и заказа билетов, например, parter.ru, ticketland.ru.
Стиль монолитное приложение не подходит потому, что…
Стиль встраиваемая система не подходит потому, что…

4.3. Описание вспомогательных интерфейсов и классов

По текстовому описанию архитектурного стиля, после обсуждения с преподавателем составить диаграмму классов с интерфейсами и классами платформы: предоставляемые компонентами в архитектуре, вспомогательными библиотеками.

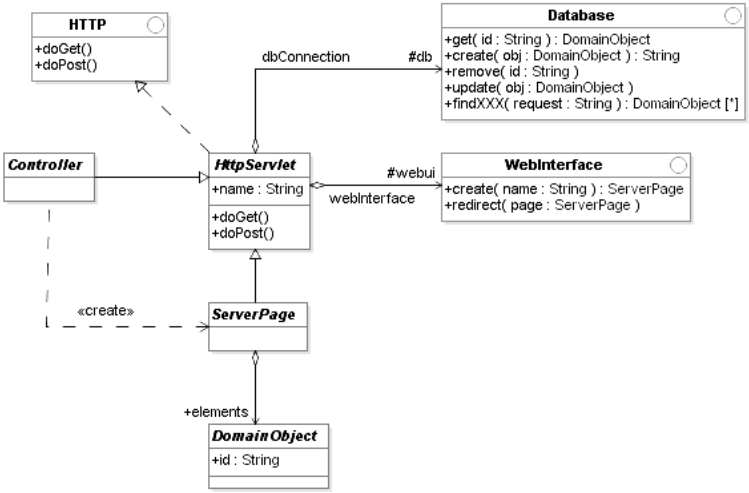

Рис. 8. Описание классов и интерфейсов платформы реализации

После диаграммы привести таблицу с перечисление классов и интерфейсов, их обязанностей и коллег.

Таблица 4. Обязанности и коллеги классов предметной области бронирования билетов.

Класс	***Контракт***	***Коллеги***
Controller	*Реализует операции, вызываемые клиентом.* *Использует WebInterface для формирования страниц пользовательского интерфейса.* *Использует Database для доступа к сохраненным объектам предметной области.*	*WebInterface* *Database* *ServerPage*
ServerPage	*Хранит объекты предметной области для отображения клиенту*	*DomainObject*
…		

4.4. Структура компонентов системы и решения по реализации архитектуры

Привести диаграмму компонентов согласно архитектурному стилю. Показать на диаграмме предоставляемые и используемые интерфейсы, соединители.

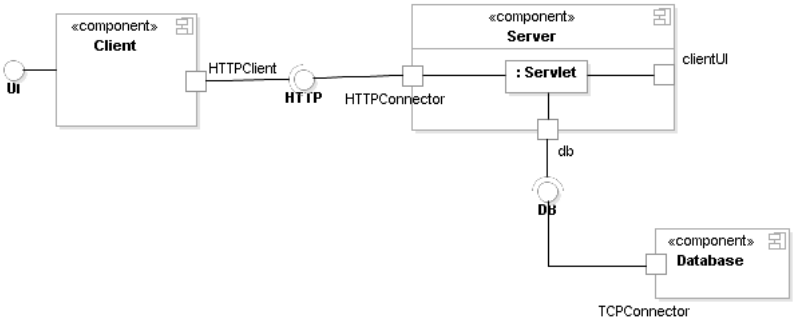

Рис. 9. Структура системы

В таблице указать назначение каждого компонента. Заполняется по описанию архитектуры.

Таблица 5. Обязанности и коллеги классов предметной области бронирования билетов.

Компонент	***Назначение***	***Интерфейсы и порты***
Client	*Предоставляет графический интерфейс пользователям.* *Взаимодействует с сервером*	*UI* *HttpClient:~HTTP*
Server	*Предоставляет HTTP интерфейс, принимает запросы на обработку от клиента.* *Взаимодействует с базой данных.* *Содержит сервлеты, реализуемые приложением.*	*HTTPConnector:HTTP* *clientUI:ClientUI* *db:~DB*
Database	…	…

4.5. Структура пакетов

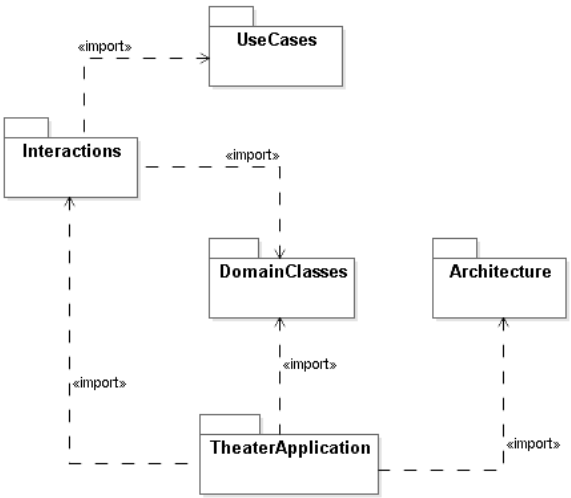

Рис. 10. Структура пакетов модели системы

Привести диаграмму пакетов и зависимостей между ними. Диаграмма пакета должна отражать структуру проекта в среде моделирования. Привести обоснование распределения элементов модели по пакетам с использованием принципов проектирования.

4.6. Размещение компонентов системы

Привести диаграмму размещения, показать узлы, указать артефакты. Показать, как артефакты реализуют пакеты приложения. Разместить компоненты по узлам. Исходя из описания архитектурного стиля, пояснить принятые решения.

5. Описание механизмов реализации

В данном разделе по аналогии с первым заданием нужно провести анализ взаимодействий, описанных в вариантах использования, и механизмов инициализации и завершения работы системы. Для них нужно проработать, какие классы будут участвовать в реализации, и какие обязанности они будут иметь.

Привести диаграмму классов, на которой показать, как варианты использования реализованы кооперациями. В подразделах рассмотреть каждый ВИ в отдельности, показать сценарий на диаграмме одного из типов:

- Последовательности для описания интерактивного взаимодействия.
- Деятельность для описания координированного использования нескольких объектов, или для описания процесса с несколькими участниками.
- Схемы состояний для описания жизненного цикл объекта, описание процесса (через воплощение).

5.1. Механизмы инициализации сервера

В данном разделе описать структуру компонента, создаваемую при загрузке и инициализации экземпляра компонента.

Привести диаграмму экземпляров с заполненными слотами.

При наличии действий, выполняемых при инициализации экземпляров указать порядок их вызова и вызывающие действия элементы модели. Для этого использовать диаграмму последовательности, создание экземпляров показать сообщениями «create».

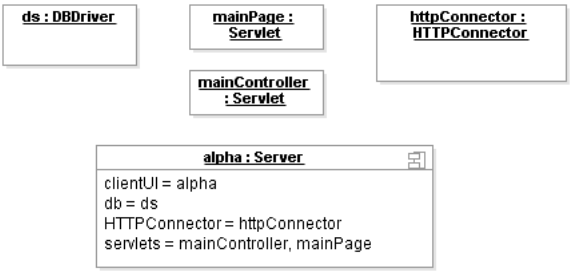

Рис. 11. Экземпляры, необходимые для инициализации сервера

5.2. Механизмы завершения работы компонента

При необходимости, указать действия, выполняемые при завершении работы компонента, порядок уничтожения частей компонента. Для этого использовать диаграмму последовательности, уничтожение экземпляров показать сообщениями «delete».

5.3. Реализация варианта использования бронировать билет

Использовать классы структуры системы из раздела 4.4, классы предметной области или их производные из раздела 2 и новые классы, которые потребуются для назначения обязанностей.

Для основного и альтернативных сценариев выделить и назначить обязанности, затем построить диаграммы поведения: последовательности, схемы состояний или деятельности.

Можно использовать карточки CRC, используя в качестве заготовок модель предметной области и платформу реализации. Для обоснования назначения обязанностей классам следует использовать эвристики GRASP (Information Expert, Creator, Controller, Low Coupling, High Cohesion, Polymorphism, Pure Fabrication, Indirection, Protected Variations) и общие принципы проектирования (OCP, LSP, SRP/ISP, DIP).

Привести таблицу с описанием обязанностей классов для данного сценария. Показать, какие паттерны каким образом применяются.

После этого, показать во фрагменте диаграммы классов интерфейсы, ассоциации, атрибуты и операции, реализующие выделенные обязанности. После диаграммы в тексте дать обоснование.

5.4. Реализация варианта использования Б

Аналогично 5.3.

5.5. Описание собственного поведения объектов класса А

В некоторых случаях объекты предметной области обладают собственным поведением. Например, классов лифт, переезд. Для описания их поведения следует составить схему состояний.

Если в первом задании диаграмма была уже составлена, здесь нужно ее уточнить с учетом выбранной архитектуры и новой структуры классов (раздел 7).

На данном этапе диаграмма служит для пояснения и прояснения поведения. Важно показать состояния, события и условия переходов.

Здесь нужно привести схему состояний, дать пояснения в тексте: обосновать, почему выделено каждое состояния, переходы между состояниями.

5.6. Описание алгоритма или процесса Б

В других случаях для описания предметной области может быть важно показать алгоритм выполнения какого-либо действия или процесса. Для этого можно использовать диаграммы деятельности.

Если в первом задании диаграмма была уже составлена, здесь нужно ее уточнить с учетом выбранной архитектуры и новой структуры классов (раздел 7).

На диаграмме обозначить основные шаги процесса, последовательность действий, зависимости по управлению. Для назначения действий участникам (акторам, классам) рекомендуется использовать разделы (плавательные дорожки).

Здесь привести диаграмму деятельности, дать пояснения по действиям и разделам.

6. Обеспечение нефункциональных требований

Перечислить в подразделах нефункциональные требования, упоминаемые в постановке задачи. Пояснить, каким образом они удовлетворены в архитектуре и дизайне системы.

6.1. Доступность системы через Интернет

В выбранной трехзвенной архитектуре слои взаимодействуют посредством сетевых протоколов, используемых в Интернет. Поэтому слой представления (клиент) можно разместить на персональном компьютере пользователя, а слои бизнес-логики и базы данных – в центре обработки данных хостинг-провайдера.

6.2. Отображение интерфейса пользователя в веб-браузере

Для реализации слоя бизнес логики в выбранной архитектуре будет использован веб-сервер. Модули веб-сервера (точнее – контейнера сервлетов), реализуемые приложением уточняют класс сервлет, который предназначен для формирования ответов на запросы клиентов. В системе определен специальный вид модулей – серверные страницы, которые формируют HTML страницу с описанием указанных им объектов предметной области. Сформированный HTML передается клиенту для отображения в веб-браузере.

6.3. Отображение упрощенной версии интерфейса пользователя для мобильного браузера

Для доступа к системе из мобильного браузера используется другой адрес URL. Поэтому запросы к системе с обычного веб-браузера обрабатываются одним контроллером, а с мобильного – другим контроллерам, имеющим другой адрес. Контроллер для доступа с мобильного браузера использует отдельный набор страниц для формирования упрощенного HTML для отображения в мобильном браузере.

7. Описание логической структуры и реализация системы

В данном разделе описывается модель системы так, как она будет реализована. Модель показать на диаграмме классов, сгруппировать по всем механизмам обязанности классов и привести их в таблице в разделе 7.1. Пояснить, каким образом обязанности отображаются в операции и атрибуты классов.

В последующих разделах привести заготовки реализации классов. Список классов и операций для реализации согласуется с преподавателем:

- Реализовать операции для основных сценариев.
- Реализовать конструкторы / деструкторы.

7.1. Логическая структура системы

Привести одну или несколько диаграмм классов для представления различных аспектов системы: структура страниц, классы предметной области и т. д. По сути это одна модель, представленная по частям для удобства.

Привести таблицу с описанием обязанностей классов, собранных из таблиц для кооперации (разделы 5.x), в которых класс участвует. Пояснить, каким образом объединены разные обязанности.

7.2. Реализация класса Performance

Привести реализации операций класса, список которых согласован с преподавателем. Использовать один из следующих языков:

- Groovy, JavaScript.
- Псевдокод.
- C++, Java, C# - нужно показать, в какие типы языка отображены типы UML.

Листинг 1. Реализация класса Performance

```
List<Ticket> availableTickets;
List<Ticket> bookedTickets;

private Ticket chooseTicket() {
	if (availableTickets.size() > 0) {
		return availableTickets.get(0);
	}
}

public Ticket bookTicket() {
	Ticket ticket = chooseTicket();

	availableTickets.remove(ticket);
	bookedTickets.add(ticket);
}
```

7.3. Реализация класса Б

Аналогично 7.2.

8. Приложение 1. Артефакты проектирования

В приложении нужно привести фотографии или копии артефактов, использованных при разработке, которые подтверждают применение методов проектирования.

Хорошим примером являются карточки CRC, обсуждения на бумаге проектных решений.

ЛИТЕРАТУРА

1. *Фаулер М.* UML. Основы. Краткое руководство по стандартному языку объектного моделирования. – М.: Символ-Плюс, 2011. – 192 с.
2. *Ларман К.* Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку. – М.: Вильямс, 2009. – 736 с.
3. *Гамма Э., Хелм Р., Джонсон Р.* Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.:Питер, 2001. – 368 с.
4. *Басс Л., Клементс П., Кацман Р.* Архитектура программного обеспечения на практике. 2-е изд. – СПб.:Питер, 2006. – 576 с.
5. *Амблер С.* Гибкие технологии: экстремальное программирование и унифицированный процесс разработки. – СПб.: Питер, 2005. – 412 с.

Методическое издание

КОМАНДНЫЙ ПРОЕКТ ПО ОБЪЕКТНО-ОРИЕНТИРОВАННОМУ ПРОЕКТИРОВАНИЮ

Методическое пособие

Хританков *Антон Сергеевич*

Редактор *Л.В. Себова.* Корректор *О.П. Котова*

Подписано в печать 10.14.2012. Формат $60 \times 84\ ^{1}/_{16}$. Усл. печ. л. 1,75.

Уч.-изд. л. 1,5. Тираж 100 экз. Заказ №

Федеральное государственное образовательное автономное учреждение
высшего профессионального образования
«Московский физико-технический институт (государственный университет)»
141700, Московская обл., г. Долгопрудный, Институтский пер., 9

Отдел оперативной полиграфии «Физтех-полиграф»
141700, Московская обл., г. Долгопрудный, Институтский пер., 9