­­­­

Задание №1 по курсу

«Проектирование Программных Систем»

*<ИнПлан>*

Выполнили:

Синотов Илья, 191

Давыдов Виталий, 193

Содержание

TOC \o 2-2

Состав первого задания PAGEREF \_Toc \h 4

Описание задачи PAGEREF \_Toc1 \h 4

Идентификация классов PAGEREF \_Toc2 \h 4

Предварительная логическая модель выбора плана обучения PAGEREF \_Toc3 \h 7

Обновленная логическая модель бронирования билетов PAGEREF \_Toc4 \h 7

Модель вариантов использования PAGEREF \_Toc5 \h 9

Вариант использования Set mark. PAGEREF \_Toc6 \h 10

Вариант использования Search course. PAGEREF \_Toc7 \h 10

Вариант использования Choose course. PAGEREF \_Toc8 \h 10

Вариант использования Get report. PAGEREF \_Toc9 \h 11

3.7.Вариант использования Recharge PAGEREF \_Toc10 \h 11

Реализация варианта использования Set mark PAGEREF \_Toc11 \h 12

Реализация варианта использования Create courses set PAGEREF \_Toc12 \h 12

Реализация варианта использования Get report PAGEREF \_Toc13 \h 13

Реализация варианта использования Search course. PAGEREF \_Toc14 \h 13

Обязанности класса CourseSet реализованный методом +find() : CourseSet. Который возвращает список курсов удовлетворяющих критериям. PAGEREF \_Toc15 \h 14

Реализация варианта использования Choose course. PAGEREF \_Toc16 \h 14

Диаграммы последовательностей и диаграмма деятельностей для каждого варианта использования PAGEREF \_Toc17 \h 14

# Постановка задачи

## Состав первого задания

Первое задание состоит из следующих этапов:

1. Составление диаграмм вариантов использования, диаграммы классов и схемы состояний.
2. Описание основных вариантов использования как в текстовом виде, так и в виде деятельности.
3. Описать артефакты разработки (списки потенциальных классов, CRC карточки, промежуточные варианты диаграмм).

## Описание задачи

Система автоматизации индивидуальных планов. Каждый курс имеет требования, кредиты и относится к направлению. Студенты могут выбирать себе курсы. Для получения специализации нужно прослушать определенный набор базовых курсов по направлениям специализации и несколько дополнительных на общую сумму кредитов.

Все курсы платные, студент может расходовать свой бюджет кредитов на разные курсы. Поэтому должен выбирать на какие курсы ходить, чтобы получить специализацию. Деканат может составлять типовые программы. Профессора могут ставить оценки, студенты отслеживать свою успеваемость, деканат получать отчеты об успеваемости по направлениям и специализациям.

# Описание предметной области

## Идентификация классов

Для идентификации классов и построения первого варианта модели предметной области воспользуемся методом Аббота. Потенциальные классы приведены в таблице 1.

Система автоматизации индивидуальных планов. Каждый курс имеет требования, кредиты и относится к направлению. Студенты могут выбирать себе курсы. Для получения специализации нужно прослушать определенный набор базовых курсов по направлениям специализации и несколько дополнительных на общую сумму кредитов.

Все курсы платные, студент может расходовать свой бюджет кредитов на разные курсы. Поэтому должен выбирать на какие курсы ходить, чтобы получить специализацию. Деканат может составлять типовые программы. Профессора могут ставить оценки, студенты отслеживать свою успеваемость, деканат получать отчеты об успеваемости по направлениям и специализациям.

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** |
| Курс | СИАУТ | Требования, стоимость, список профессоров (преподающих этот курс), свое название. |  |
| Cтудент | СИAОУТ | Свой бюджет, список курсов, оценки по курсам, специализация, ФИО. | Просмотр учебного плана, просмотр курсов, запись на курсы (или выбор плана обучения), смена преподавателя в рамках курса. |
| Специализация | СAУТ | Название специализации, список базовых курсов, список дополнительных курсов. |  |
| Профессора |  |  | Ставить оценки. |
| Деканат | О |  | Получает отчеты об успеваемости, составляет набор типовых курсов. |
| Отчет | CИА | Имя направления, список студентов. |  |

Потенциальные классы Специализация и Направления были объединены в один класс, так как по сути являются синонимами.

Будем предполагать, что набор базовых (эмперических) курсов в специальности остается неизменным с момента создания специализации.

Классы Профессор Деканат были исключены из модели, так как они реализуются чтением и изменением хранимой информации. В дальнейшем классы Профессор и Деканат будут представлены как акторы, использующие систему.

Класс Студент так же будет представлен как актор.

Класс Отчет был создан для получения деканатом отчетов об успеваемости студентов в рамках направления.

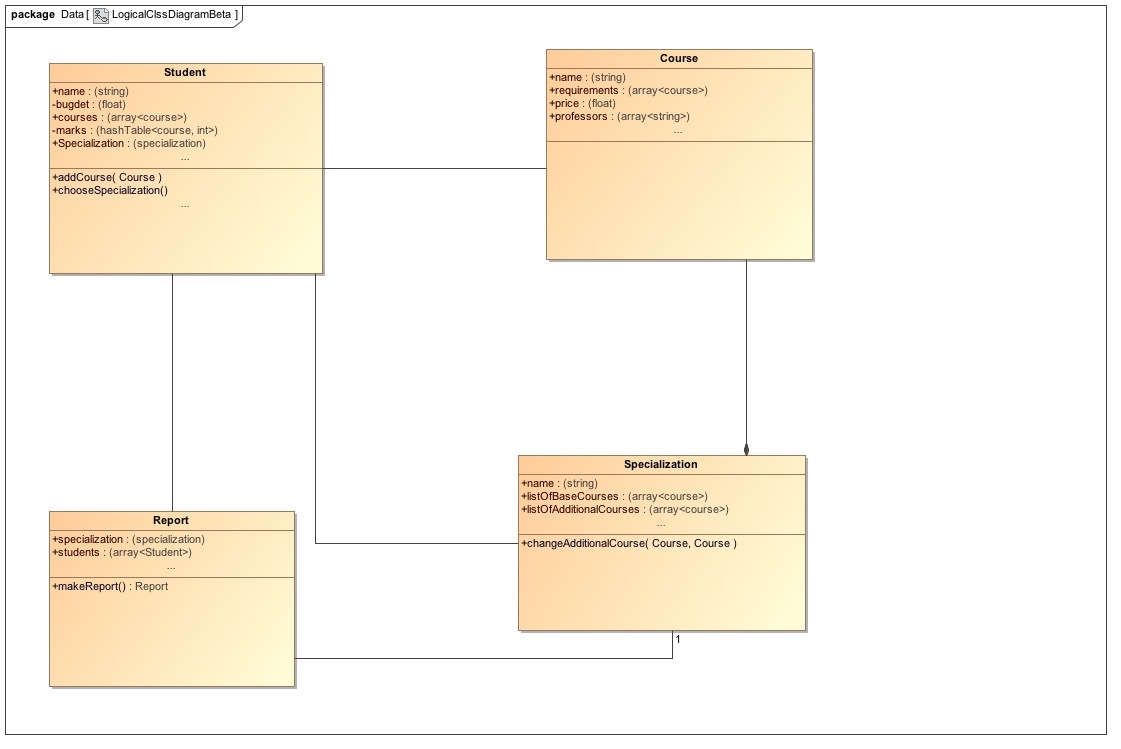
## Предварительная логическая модель выбора плана обучения

Между классами Specialisation и Course добавлено отношение композиции, так как по сути специализация и есть определенный набор базовых курсов, который прошел студент. То есть жестко с ним связана.

Связи класса Report с классами Student и Specialisation ясна по его назначению – класс Report есть отчет, который получает деканат по поводу успеваемости студетов, т.е. в классе Report у нас есть направление, по которому нужно получить отчет и список студентов, обучающихся на нем.

Диаграмма классов модели приведена на рис. 2.

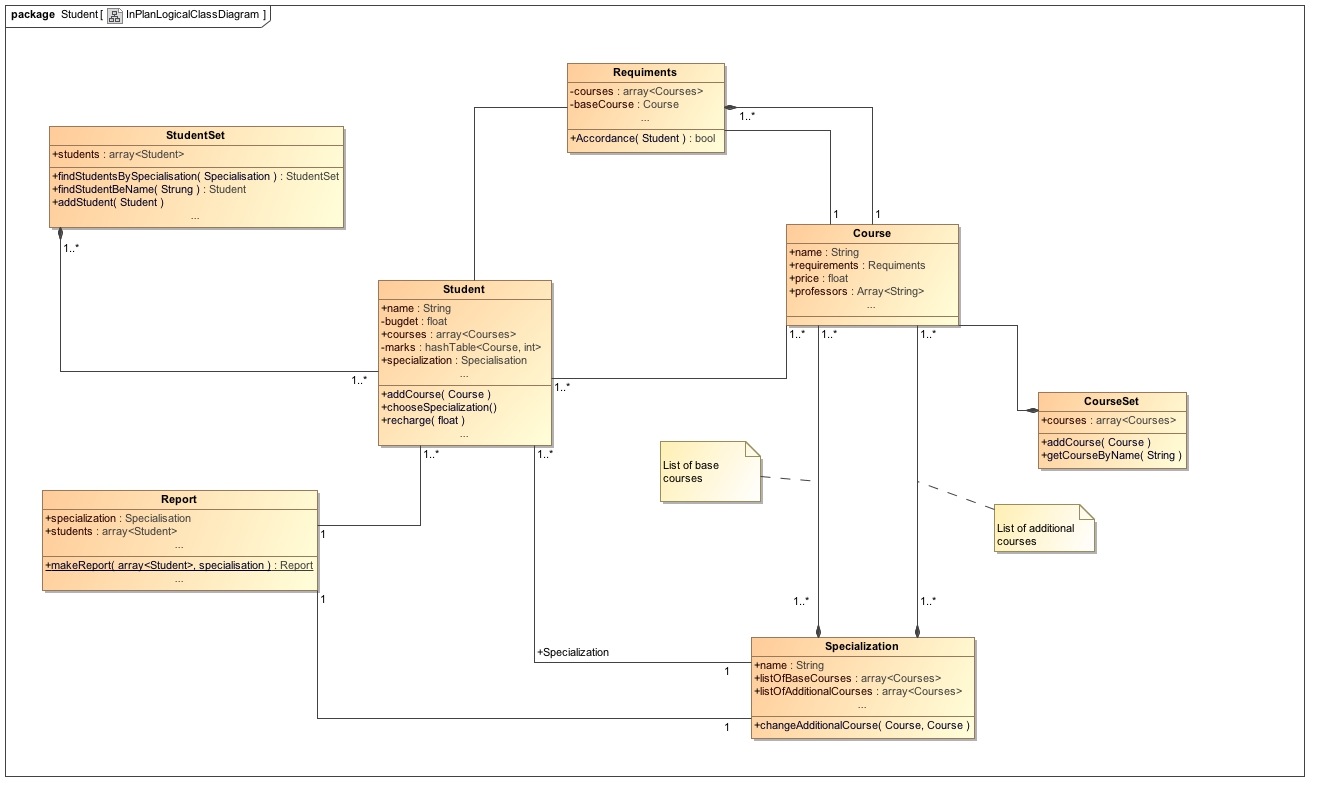
Рис. 1. Предварительная диаграмма классов выбора плана обучения



## Обновленная логическая модель бронирования билетов

Рис. 2. Обновленная диаграмма классов выбора плана обучения

Анализ взаимодействия во многом был показан на Рис. 1. В качестве финального анализа были продуманы кратности связей классов, например, связь между классами Student-Course будет типа «многие ко многим», т.к. каждый студент ходит на нескольколько курсов, при этом каждый курс посещает несколько студентов. Остальные кратности связей строились по такому же принципу.

Так же добавлена двойная ассоциация между классами Specialisation и Course, т.к. мы предполагаем, что у нас есть как набор обязательных (необходимых) базовых курсов для получения специализации, а так же набор дополнительных курсов. Прохождение обязательных и дополнительных курсов – достаточное условие для получения специализации (под базовыми курсами имеются в виду курсы общие для всех факультетов на данном курсе, например курс Общей физики в МФТИ; под дополнительными курсами будем иметь в виду все остальные курсы на факультете, не входящие в базовые, например курс Проектирования Програмных систем, который будет дополнительным по отношению к базовым на ФИВТе).

Так же метод +makeReport(array<Student>, Specialisation) : Report сделан статическим (мы просим класс Report создать объект Report c заданными параметрами).

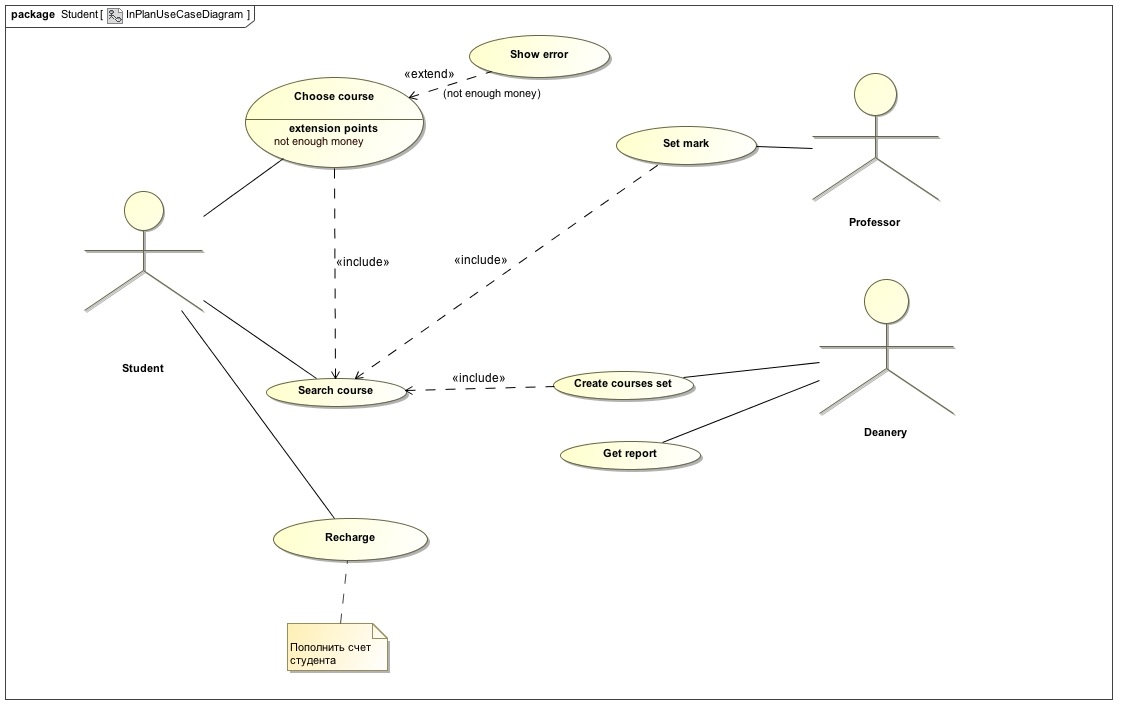
Для облегчения понимания были добавлены два поясняющих комментария.

Классы Requiments, CourseSet, StudentSet были выявлены позднее в процессе анализа реализации вариантов использования.

# Описание требований к системе

## Модель вариантов использования

В данной модели акторами являются Student, Professor, Deanery.

Студент может искать курсы и выбирать для себя подходящий. При выборе курса добавлена зависимость <<extend>> так, как у студента может не хватить бюджета на данный курс, после этого будет выдана ошибка.

Профессор может выставлять студентам оценки по своему курсу, здесь добавлена зависимость с <<include>>.

Деканат может получать отчет и составлять набор необходимых курсов.

Рис. 3. Диаграмма вариантов использования.

## Вариант использования Set mark.

**Акторы:** Professor (Профессор)

**Цель:** Профессор ставит оценку студентам, посещающим его курс.

**Предусловия:** Профессор ведет курс.

**Постусловия:** Профессор проставил оценки.

**Основной сценарий:**

1. Профессор запрашивает список курсов (дополнительно используется вариaнт использования Search course), которые он ведет. Система показывает список курсов и студентов.
2. Профессор выбирает курс и студента.
3. Ставит ему оценку.

## Вариант использования Search course.

**Акторы:** Student (Студент), Professor (Профессор), Deanery (Деканат)

**Цель:** Актор хочет найти курс или просмотреть список всех курсов.

**Предусловия:** Деканат решил создать набор курсов или профессор собрался выставить оценку студенту по курсу или студент решил найти нужный ему курс.

**Постусловия:** Актор добился основной своей цели, указанной в предусловиях.

**Основной сценарий:**

1. Актор запрашивает курс по определенным критериям поиска. Система показывает список курсов.
2. Актор выбирает курс (имеется в виду, что он не платит за него, а получает подробную информацию и описание) и использует в соответствие со своей целью.

## Вариант использования Choose course.

**Акторы:** Student (Студент)

**Цель:** Студент хочет записаться на курс.

**Предусловия:** Студент нашел курс.

**Постусловия:** Студент записался на курс.

**Основной сценарий:**

1. Студент производит оплату за курс. Система снимает деньги с бюджета студента и записывает его на участие в курсе.

**Альтернативные сценарии:**

Если на шаге 1. у Студента не хватает денег, Система уведомляет Студента, и просит его выбрать другой курс.

**3.5.Вариант использования Create сourses set.**

**Акторы:** Deanery (Деканат)

**Цель:** Деканат хочет создать набор курсов в рамках специализации.

**Предусловия:** Существование деканата.

**Постусловия:** Создан набор курсов и добавлен в список курсов.

**Основной сценарий:**

1. Деканат составляет список курсов (для этого дополнительно используется вариант использования Search course).
2. Деканат отправляет список. Система добавляет курсы в базу.

## Вариант использования Get report.

**Акторы:** Deanery (Деканат)

**Цель:** Деканат хочет получить отчет об успеваемости студентов в рамках направления.

**Предусловия:** Есть список курсов и студентов.

**Постусловия:** Отчет создан и получен.

**Основной сценарий:**

1. Деканат отправляет запрос с информацией о том, по какой специализации получить отчет.
2. Деканат получает отчет.

## 3.7.Вариант использования Recharge

**Акторы:** Student (Студент)

**Цель:** Студент хочет пополнить свой счет (Budget) в системе.

**Предусловия:** У студента есть банковский счет.

**Постусловия:** Студент пополнил счет.

**Основной сценарий:**

1. Отправка запроса в банк на перевод денежных средств.
2. Ожидаем подтверждение из банка о корректности запроса.
3. Увеличивается значение переменной Budget у Student.

**Альтернативные сценарии:**

Если на шаге 2. не получили подверждение из банка, выводим ошибку.

# Моделирование взаимодействий и поведения

## Реализация варианта использования Set mark

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

Таблица 2. Обязанности и коллеги участников реализации варианта использования Set mark.

|  |  |  |
| --- | --- | --- |
| **Объект класса** | **Контракт** | **Коллеги** |
| Professor | Использует курсы  Использует студентов  Вызывает представление | Course  Student |
| CourseSet | Хранит список курсов | Course |
| StudentSet | Хранит список студентов | Student |

Обязанность класса Professor использовать курсы, реализуется по предположению, что профессор знает какие курсы он ведет, но не знает всех студентов, посещающих его курс.

Обязанности классов CourseSet и StudentSet по хранению информации реализуются простыми геттарами и по сути являются просто контейнерами для хранения нужных курсов.

## Реализация варианта использования Create courses set

Результаты анализе основного сценария с помощью карточек CRC приведены в таблице 3.

Таблица 3. Обязанности и коллеги участников реализации варианта использования Create courses set.

|  |  |  |
| --- | --- | --- |
| **Объект класса** | **Контракт** | **Коллеги** |
| Deanery | Использует курсы  Использует специализацию | Course  Specialisation |
| CourseSet | Хранит список курсов | Course |
|  | | |

Обязанности классов CourseSet по хранению информации (и добавлению информации) реализуется геттерами и операцией +addCourse(Course) которая добавляет какой-то курс в контейнер.

## Реализация варианта использования Get report

Результаты анализе основного сценария с помощью карточек CRC приведены в таблице 4.

Таблица 4. Обязанности и коллеги участников реализации варианта использования Get report.

|  |  |  |
| --- | --- | --- |
| **Объект класса** | **Контракт** | **Коллеги** |
| Deanery | Использует отчеты | Report |
| Report | Хранит специализацию  Хранит студентов | Specialisation  Student |
|  | | |

Обязанности классов Report по хранению специалиализации и списка студентов, по которым получить отчет реализуется стандартными геттерами и сеттерами, а так же операцией +getReport() : Report которая возвращает готовый отчет.

## Реализация варианта использования Search course.

|  |  |  |
| --- | --- | --- |
| **Объект класса** | **Контракт** | **Коллеги** |
| Deanery | Использует поиск | Course |
| CourseSet | Хранит список курсов.  Реализован метод, для отображения курсов удовлетворяющих критериям | Specialisation  Course |
| Student | Использует поиск | Course |
| Professor | Использует поиск | Course |

## Обязанности класса CourseSet реализованный методом +find() : CourseSet. Который возвращает список курсов удовлетворяющих критериям.

## Реализация варианта использования Choose course.

|  |  |  |
| --- | --- | --- |
| **Объект класса** | **Контракт** | **Коллеги** |
| Student | Выбирает курс. | Requirements |
| Requirements | Хранит требования для курса.  Реализован метод, проверки студента на соответствие требованиям. | Student |

Обязанности класса Requirements реализованы стандартными сеттерами и геттерами. А так же методом +Accordance(Student): bool , который проверяет студента на соответствие требованиям курса.

В дополнение к этому у класса Student есть метод +addCourse : (Course) , которая записывает студента на курс или выводит сообщение об ошибке, и использует функцию Compare , как вспомогательную.

## Диаграммы последовательностей и диаграмма деятельностей для каждого варианта использования

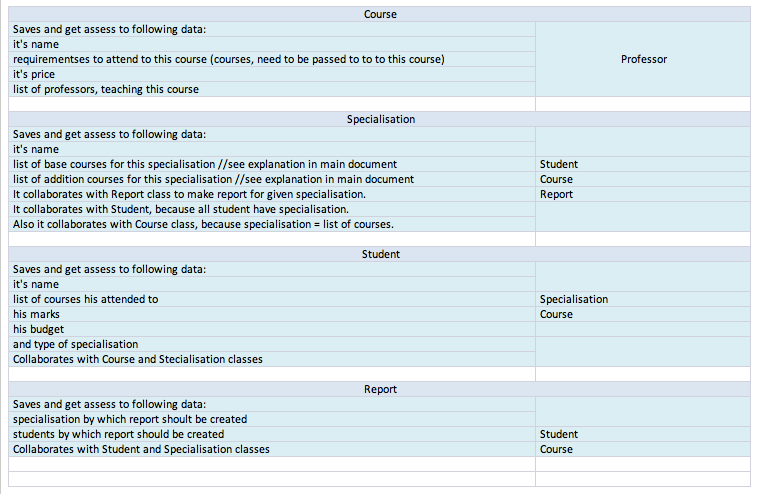
Чтобы лучше показать как будут реализованы use case'ы были построены диаграммы деятельности для каждого варианта использования в отдельности (см. проектный файл в MagicDraw).

Так же была сделана дииаграмма деятельности для более детаного показа работы программы (см. проектный файл в MagicDraw).

# Приложение 1. Артефакты проектирования

Т.к. никаких разработок на бумаге не было, мы решили использовать в качестве артефактов проектирования CRC карточки, а так же историю коммитов в системе git.

**Рис 5. CRC карточки**



**Рис. 6 часть истории git коммитов**

