Лабораторная работа 3

Тема. Классы. Диаграммы классов в языке UML

# Теоретическое введение. Язык UML

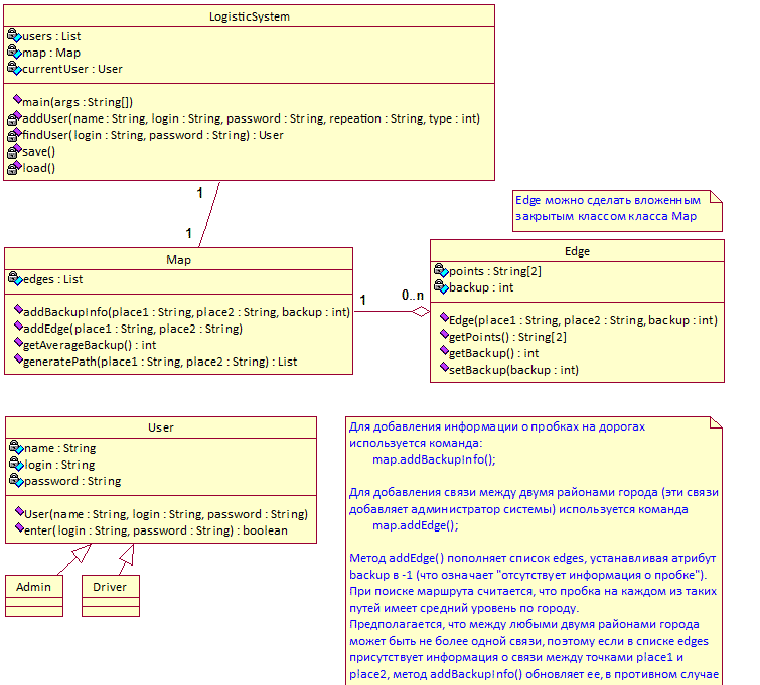
Внимание! На данном этапе обучения вы уже должны владеть знаниями по этой теме. Если их нет, а материалы для повторения непонятны или недостаточны, вы с заданиями не справитесь! Необходимо обратиться к литературе по данной теме.

На этапе проектирования программного продукта, разрабатываемого в соответствии с объектно-ориентированной методологией, составляется диаграмма классов. Одним из стандартных средств для создания этих диаграмм является язык UML.

Обязанности разработчиков могут разделяться. Проектировщики составляют диаграммы классов, а программисты реализуют эти диаграммы в программном коде. Для того, чтобы между программистами и проектировщиками не было разночтений, и те и другие должны знать UML.

Основное задание практикума по Java содержит уже разработанные фрагменты UML-диаграмм (чтобы студенты могли сосредоточиться на программировании). Необходимо уметь в них разбираться.

Напомним основные обозначения, применяемые в диаграммах классов на примере UML-диаграммы системы LogisticSystem (см. задание 10).



Классы изображаются на диаграмме прямоугольниками, разделенными на три части. В верхней части указывается имя класса. В средней части перечисляются атрибуты (поля) класса, а в нижней — его методы. На рисунке изображены классы User, Admin, Driver, Map, Edge, LogisticSystem.

Закрытые (private) члены класса помечаются знаком минус или изображением замочка. На рисунке в классе LogisticSystem все поля и методы закрыты, кроме метода main(). У других классов закрытыми являются все поля (в соответствии с принципом инкапсуляции), а методы открыты.

Тип атрибутов (полей) класса указывается через двоеточие после его имени. Так же указываются типы параметров методов и возвращаемых значений методов. Например, в классе LogisticSystem показано описание метода findUser():

findUser(login : String, password : String) : User

На языке Java заголовок этого метода на самом деле выглядит так:

public User findUser(String login, String password);

Однако UML не привязан к конкретному языку программирования, поэтому заголовки методов придется "переводить".

Наследование изображается в виде стрелки с полым белым наконечником, указывающей от класса-потомка к классу-предку. На рисунке классы Admin и Driver являются наследниками класса User. Унаследованные члены класса в классах-потомках не отображаются.

Между классами могут быть еще два вида отношений. Ассоциация — это связь между объектами двух классов, изображаемая прямой линией. Например, между классом LogisticSystem (система составления маршрутов) и классом Map (карта города) существует ассоциативная связь. В данном случае она означает, что система пользуется картой для выполнения своих функций. Агрегация — это отношение вида "часть/целое". Например, класс Edge (дорога между районами) является частью класса Map, то есть карта состоит из районов и связей между ними (как видно из рисунка, сами районы отдельным классом не представлены, они описываются с помощью обычных строк — см. атрибут points класса Edge). Агрегация изображается в виде линии с ромбом на конце (ромб ставится около класса, являющегося частью другого).

Отношения ассоциации и агрегации помогают понять взаимосвязи между разными классами. Эти взаимосвязи могут быть уже реализованы на диаграмме через атрибуты (поля) классов. Например, в классе Map есть атрибут edges, представляющий собой список ссылок на объекты класса Edge, а в класс LogisticSystem уже включен атрибут map для ссылки на карту, используемую системой. Поэтому при выполнении заданий следует обращать внимание на ассоциации и агрегации лишь для лучшего понимания модели.

Для лучшего понимания модели могут оказаться полезными мощности отношений, которые проставляются на концах линий. Рассмотрим агрегацию между классами Map и Edge. Около класса Map стоит число 1, которое означает, что объект Edge может являться частью только одной карты (что неудивительно, поскольку в программе предусмотрена единственная карта). Около класса Edge стоит 0..n. Это означает, что с объектом класса Map может быть связано произвольное число объектов класса Edge, в том числе ни одного.

Заметим, что на диаграмме опущены две ассоциативных связи между классами LogisticSystem и User. Первая связывает систему и зарегистрированных в ней пользователей (реализуется через атрибут users). Вторая связывает систему и текущего пользователя (реализуется через атрибут currentUser). Эти связи опущены для увеличения наглядности остальной диаграммы.

**Литература по теме:**

*1. Терри Кватрани. Rational Rose 2000 и UML. Визуальное моделирование.*

# Задание Разработать приложение со структурой классов варианта

1. Реализовать классы в соответствии с предложенной диаграммой.
2. В каждый класс включите конструкторы, необходимые для заполнения атрибутов класса (реализуйте проверку переданных в конструктор параметров).
3. Рекомендации по доработке функциональности программы:
   1. В базовой версии программы все объекты, предусмотренные в ней (пользователи, заявки, вопросы тестов и т.д.) могут только создаваться, но не редактироваться. Однако реальная программа, конечно, должна содержать возможность просматривать и изменять параметры существующих объектов (например, изменить пароль пользователя, поменять заявку до того, как она была обработана и т.д.). Рекомендуется реализовать соответствующую возможность. Для этого нужно разработать интерфейс, позволяющий выбрать один из существующих объектов и изменить его, а также добавить в нужный класс методы для изменения соответствующих атрибутов.
   2. Классы, необходимые для построения пользовательского интерфейса, на предложенных диаграммах отсутствуют. В целом предложенные диаграммы не являются догмой и могут быть усовершенствованы (или вообще изменены).
   3. В каждой программе есть пользователи двух типов: администратор и клиенты. Предполагается, что администратор в программе один. Он должен быть добавлен сразу после первого запуска программы.

# Варианты задач

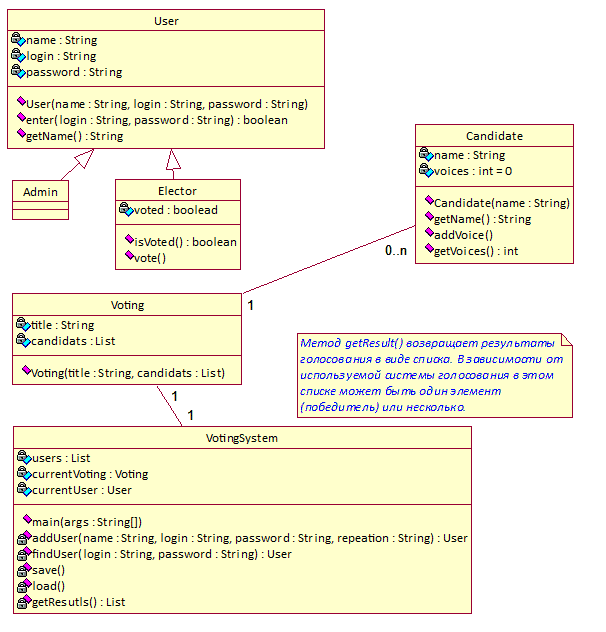
# Система электронного голосования

Написать программу для проведения выборов. Избиратели должны предварительно зарегистрироваться в системе. Администратор заполняет список кандидатов. Каждый участник (по сети) или с того же самого компьютера входит в систему и голосует. Данные о проголосовавших накапливаются в базе данных. Дважды проголосовать нельзя. По окончании периода голосования администратор запускает процедуру подсчета голосов и система выдает результат.

Основные алгоритмы

*Подведение итогов выборов.*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс для создания голосования.
* Интерфейс для голосования.
* Интерфейс для проверки результатов голосования.

Таблицы с данными

* Выборы (наименование, сроки и т.д.)
* Кандидаты Пользователи

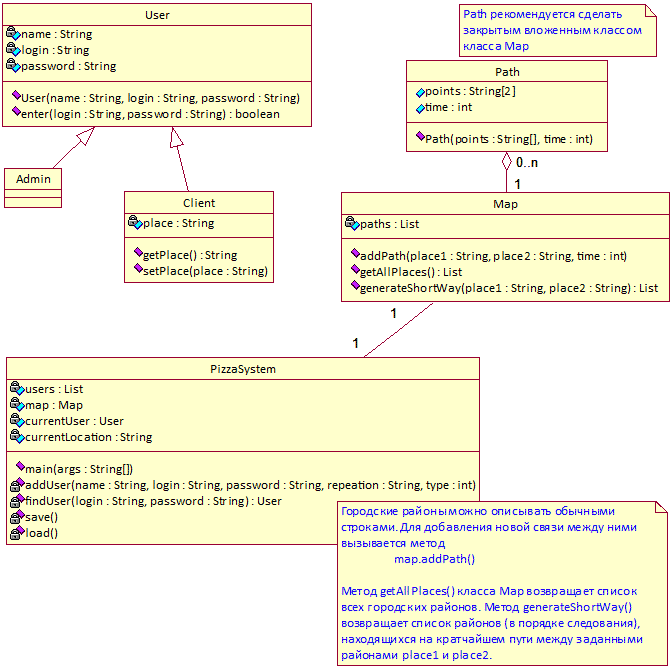
# Обработка заказов на доставку пиццы

Написать программу для составления оптимального маршрута разносчика пиццы. Пользователи, желающие заказать пиццу, оставляют в системе заявку, указывая район своего проживания. Кроме того система постоянно накапливает информацию о способах передвижения между соседними районами города. Когда поступает заказ, программа должна найти короткий путь от текущего местонахождения вагончика с пиццей до района проживания заказчика.

Основные алгоритмы

*Нахождение кратчайшего маршрута.*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для создания «карты» (на самом деле это просто граф) города.
* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс для заполнения заявки.

Интерфейс для обработки результатов.

Таблицы данных:

* Пользователи
* Карта города (районы и связи между ними)

# Ипподромный тотализатор

Написать программу для приема ставок и расчета выигрышей на скачках. Пользователи, зарегистрированные в системе, могут поставить любую сумму на одну из семи лошадей. Предварительно администратор системы вносит список лошадей. По результатам заезда (его можно смоделировать на основе случайных чисел) должна рассчитываться сумма выигрыша.

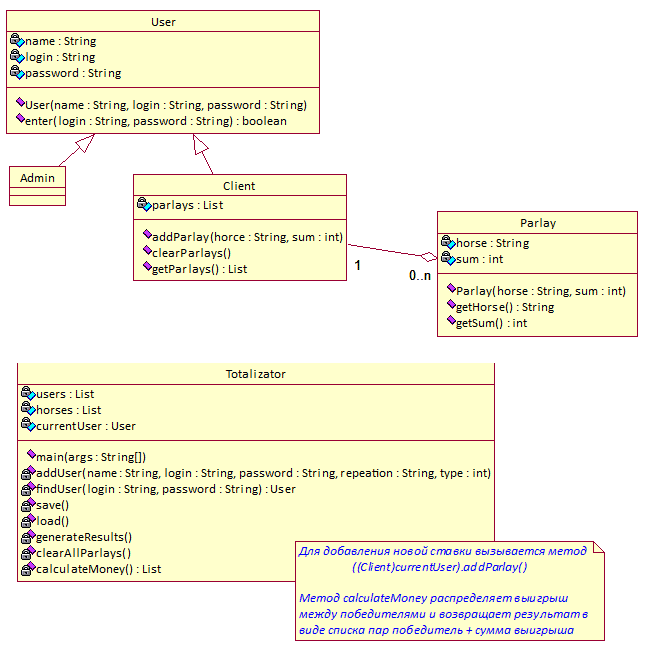
Желательно, чтобы система могла обрабатывать несколько видов ставок (какая лошадь придет первой, какая лошадь придет последней и т.д.).

Основные алгоритмы

*Моделирование результатов забега на основе случайных чисел (метод generateResults())*

*Подсчет и распределение суммы выигрыша (метод calculateMoney())*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс для описания забега.
* Интерфейс для приема ставок.
* Интерфейс для подсчета суммы выигрыша.

Таблицы данных:

* Пользователи
* Ставки
* Лошади

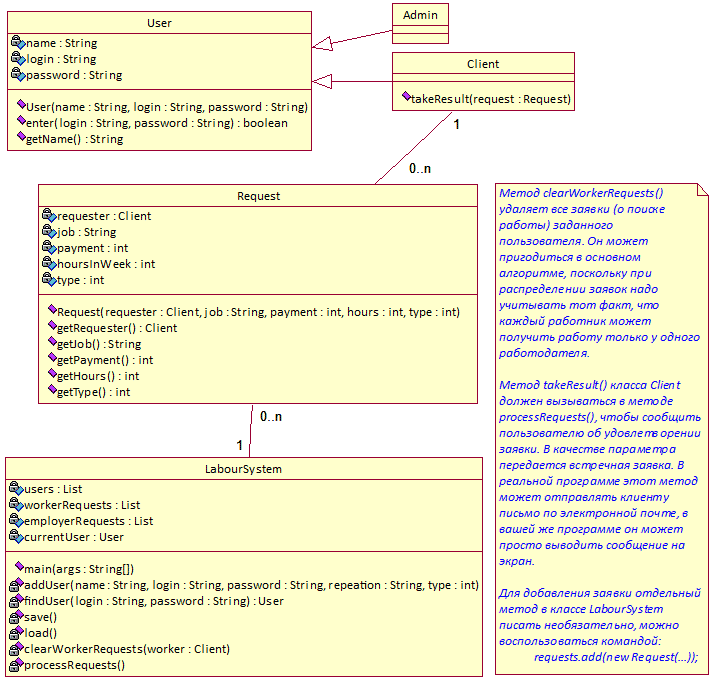
# 4истема распределения вакансий на рынке труда

Написать программу для распределения вакансий. Пользователи (работники и работодатели) регистрируются в системе, после чего могут делать заявки. Можно предложить работу (указав требования к работнику, зарплату и другие параметры), либо искать работу (указать желаемую должность, зарплату и т.д.). После накопления достаточного количества заявок система приступает к их распределению, чтобы максимально учесть пожелания пользователей и возвращает результат в виде пар работник-работодатель.

Основные алгоритмы

*Распределение заявок*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс для регистрации заявки.
* Интерфейс для распределения заявок.

Таблицы данных:

* Текущие заявки
* Пользователи

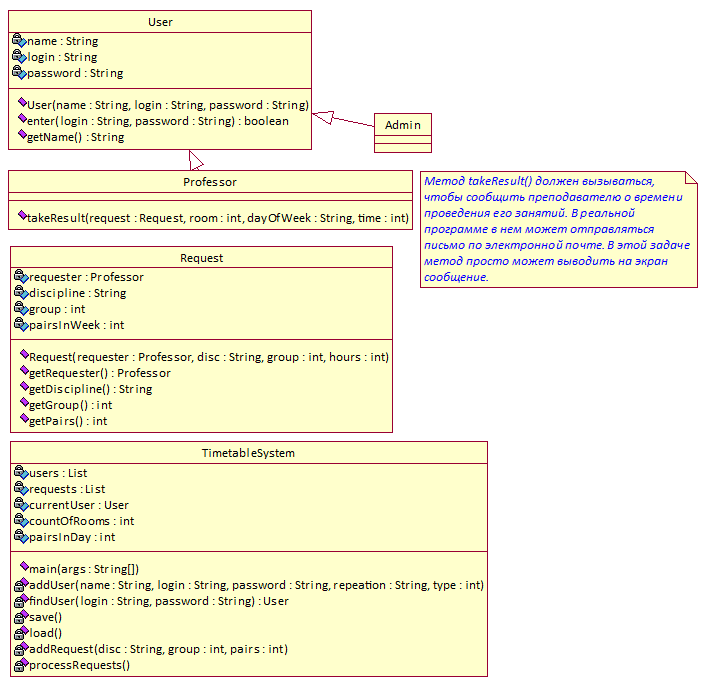
# Система автоматического составления расписания

Написать программу для автоматического составления расписания занятий. Пользователи, зарегистрированные в системе (преподаватели) делают заявки (указывают номер студенческой группы, название предмета и количество занятий в неделю). Администратор указывает количество свободных аудиторий (и другие параметры при необходимости). Накопив все заявки, система должна составить расписание. Желательно, чтобы расписание было как можно более качественным (отсутствие «дырок» у студентов и преподавателей, равномерная загруженность).

Основные алгоритмы

*Автоматическое составление расписания*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей (преподавателей).
* Интерфейс администратора (для указания числа аудиторий и др.)
* Интерфейс для приема заявок.
* Интерфейс для составления расписания.

Таблицы данных:

* Пользователи
* Заявки
* Информация об университете (число аудиторий и др.)

# Оболочка для тестирования

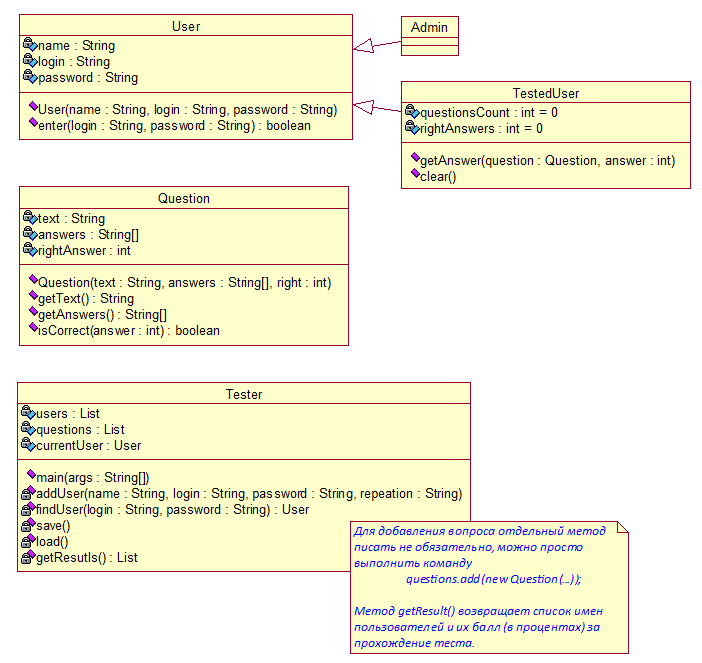
Написать программу для проведения тестирования. Администратор должен иметь возможность добавлять в программу вопросы (указывая при этом варианты ответов и отмечая правильный вариант). Пользователи проходят тестирование, отвечая на эти вопросы и получая оценку в виде процента правильных ответов.

Желательно, чтобы программа позволяла переключаться между тестами. Т.е. администратор может иметь возможность создавать новые тесты и при добавлении вопроса указывать, к какому тесту он относится, а пользователь — выбирать нужный тест из списка.

Основные алгоритмы

*Вычисление результата тестирования*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс для добавления вопросов (и тестов).
* Интерфейс для прохождения тестирования.

Таблицы данных:

* Вопросы
* Пользователи (и результаты их тестирования)

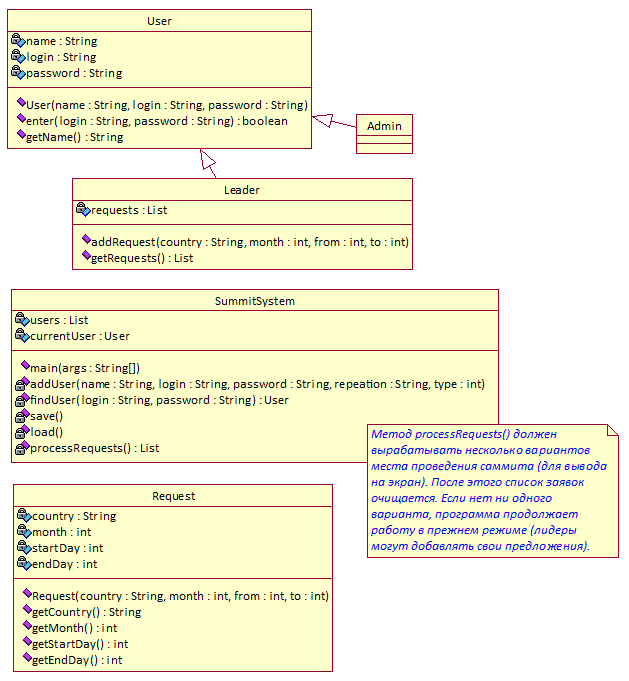
# Система для выбора места проведения саммита

Необходимо разработать эксклюзивную программу для лидеров стран, собирающихся организовать саммит. Каждый лидер может предложить несколько вариантов времени и места встречи. Место встречи — страна (может вводиться вручную или выбираться из списка уже введенных ранее стран), время задается сочетанием месяц + с какого дня + до какого дня. Система подводит итоги, определяя возможные варианты времени и места встречи, подходящие для всех лидеров стран. Если таких вариантов нет, лидеры стран могут продолжать добавлять свои варианты.

Основные алгоритмы

*Нахождение возможностей для проведения саммита.*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс для добавления варианта проведения саммита.
* Интерфейс для определения результатов.

Таблицы данных:

* Пользователи.
* Варианты мест и дат проведения саммита.

# Игра «морской бой»

Написать программу для игры в морской бой. Пользователи регистрируется в системе и указывают расположение своих кораблей. Затем пользователи по очереди пытаются «попасть» в корабли противника (вводя координаты в текстовое поле или указывая мышкой клетку на карте, что предпочтительнее). В упрощенной версии играть могут два пользователя и после каждого хода программа завершает «сеанс» связи с текущим пользователем и запрашивает пароль следующего.

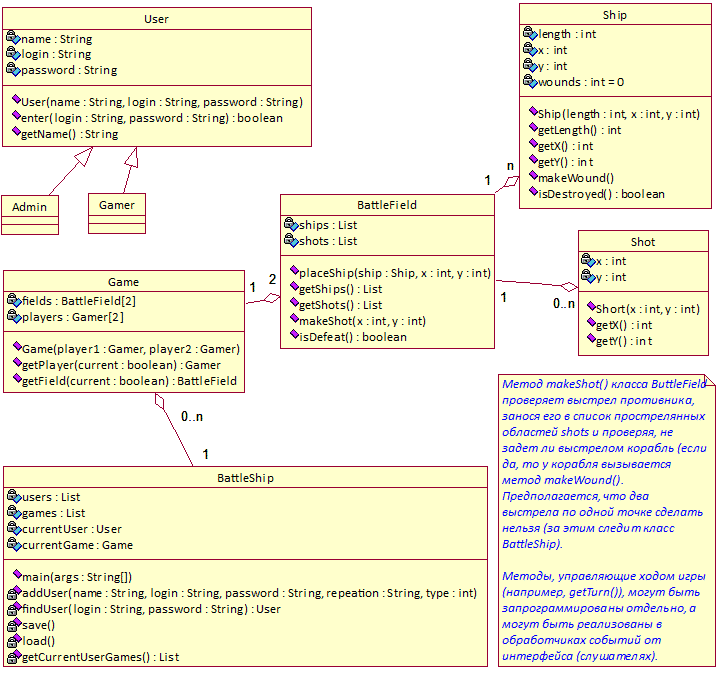
В более продвинутой версии программа может поддерживать сеанс одновременной игры с несколькими пользователями (в этом случае надо указать, чьи корабли следует бомбардировать).

В сетевой версии игры можно сделать возможность, чтобы игра не прерывалась после каждого хода, а просто блокировалась в ожидании хода противника.

Основные алгоритмы

*Проверка попадания по кораблю и проверка окончания игры.*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации и расстановки кораблей.
* Интерфейс для выстрела.

Таблицы данных:

* Игроки
* Позиции кораблей и «прострелянных» точек каждого игрока.

# Система дистанционного обучения.

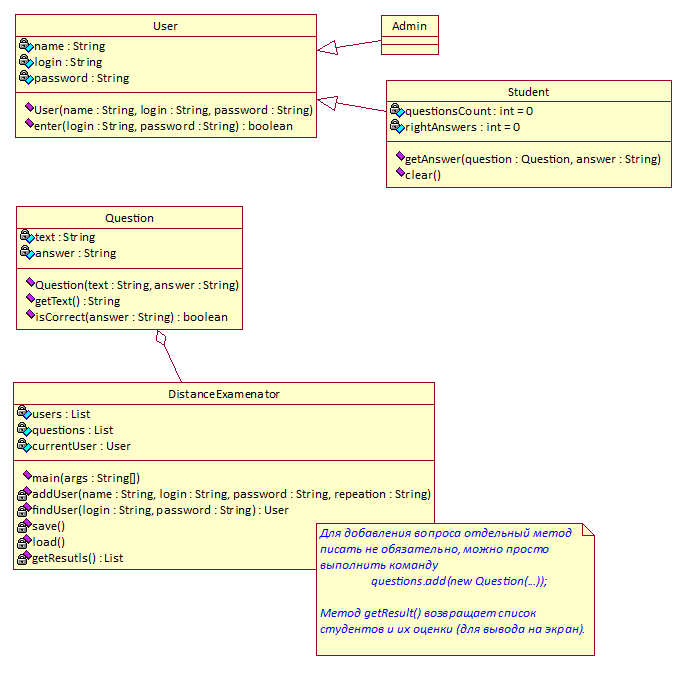
Написать программу для проведения удаленного экзамена. Преподаватель готовит список вопросов к экзамену, указывая для каждого из них правильный ответ. Студенты сдают экзамен, отвечая на пять вопросов, случайным образом выбранных системой и получают оценку.

Желательно сделать так, чтобы системой могли пользоваться несколько преподавателей. Студент после регистрации должен выбрать предмет. Если экзамен уже сдан, его в этом списке не будет.

Основные алгоритмы

*Проверка ответов и проставление оценки*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации студентов.
* Интерфейс для добавления вопросов.
* Интерфейс для сдачи экзамена.

Таблицы данных:

* Вопросы
* Студенты (и их оценки)

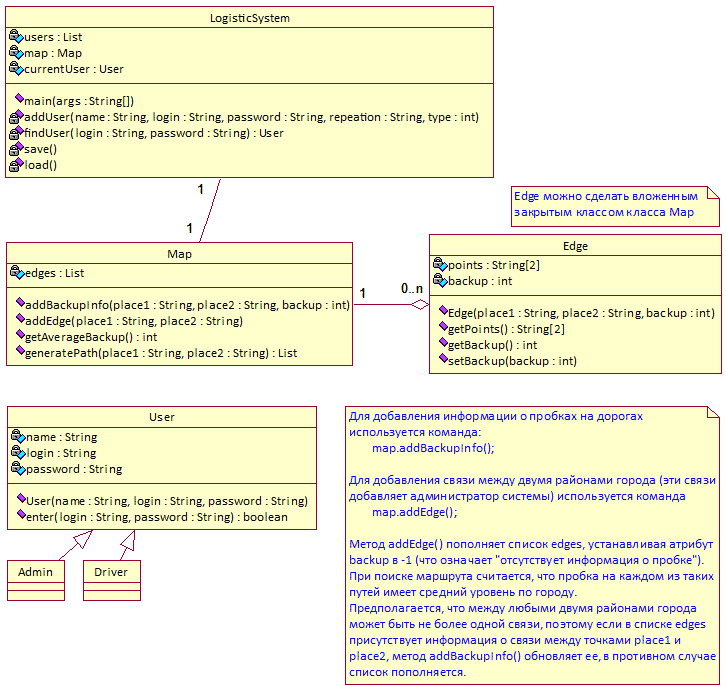
# Система контроля за пробками

Написать программу для автомобилистов, отслеживающую возникающие на дорогах пробки и предлагающую наименее загруженный путь между двумя точками. Автомобилисты могут использовать эту программу, чтобы сообщить о пробке на пути от одной точки города до соседней (этими точками могут быть смежные районы города). Каждая пробка получает оценку загруженности по десятибалльной системе. Программа также по запросу для двух точек находит маршрут, на котором общая продолжительность стояния в пробках минимальна (разницу в расстоянии между разными смежными районами мы игнорируем).

Основные алгоритмы

*Поиск пути сквозь пробки*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс для сообщения о пробке (или ее отсутствии).
* Интерфейс для запроса.
* Интерфейс для отображения составленного маршрута.

Таблицы данных:

* Пользователи
* Районы (точки)
* Связи между районами (с указанием загруженности)

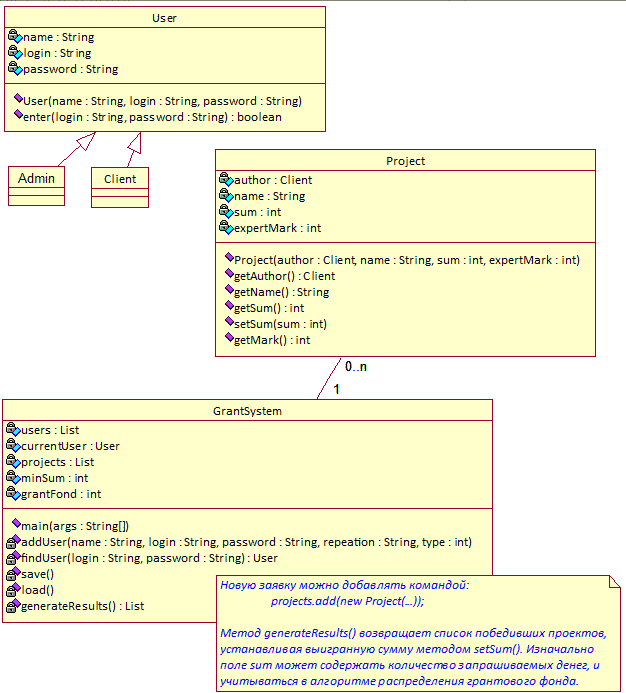
# Конкурс проектов

Написать программу для проведения конкурса. Программа должна распределять грантовый фонд конкурса между участниками в зависимости от экспертных оценок, выставленных каждому участнику. При этом сумма, выделенная каждому участнику не может быть меньше некоторой минимальной величины (участники, «заработавшие» ниже минимума, из конкурса выбывают). Заявки от участников поступают уже с проставленными экспертными оценками. Процедуру определения результатов конкурса запускает администратор (после того, как заканчивается срок приема заявок).

Основные алгоритмы

*Определение результатов конкурса*

Рекомендуемая диаграмма классов



Интерфейсы:

Интерфейс для входа в систему. Интерфейс для регистрации пользователей. Интерфейс для заполнения параметров конкурса (общий фонд и минимальная сумма). Интерфейс для подачи заявки. Интерфейс для расчета результатов.

Таблицы в базе данных:

Пользователи Параметры конкурса Заявки

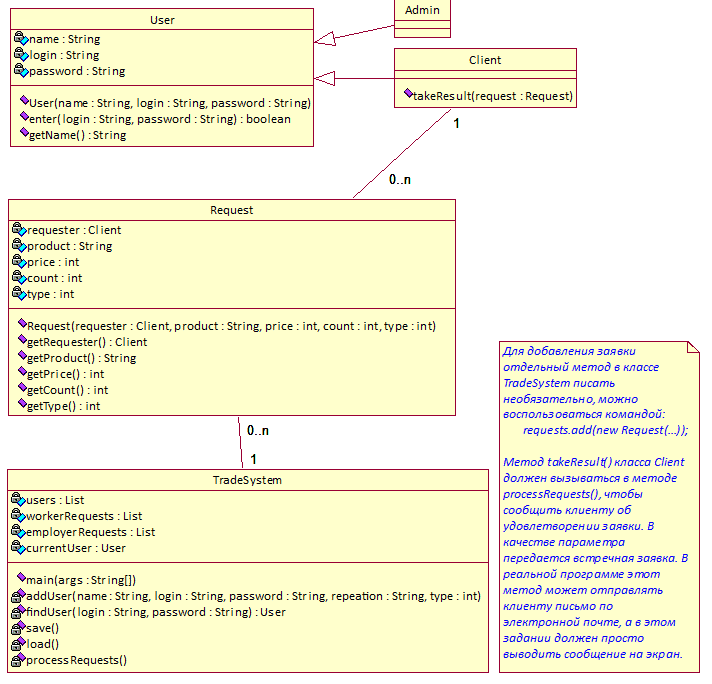
# 12\*. Бизнес-аукцион

Написать программу для проведения торгов по заявкам. Участники регистрируются в системе и подают заявки на продажу и закупку товара (кроме наименования и цены товара могут указываться дополнительные параметры, такие как качество). В конце дня система (по команде администратора) сопоставляет заявки и выдает результаты торгов, пытаясь максимально удовлетворить потребности участников. Заявки, которые остались неудовлетворены, переносятся на следующий день.

Основные алгоритмы

Расчет результатов торгов (на основе некоторого критерия оптимальности)

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации участников.
* Интерфейс для добавления заявок.
* Интерфейс для вывода результатов.

Таблицы данных:

* Участники торгов
* Заявки

# Система для координации участников проекта

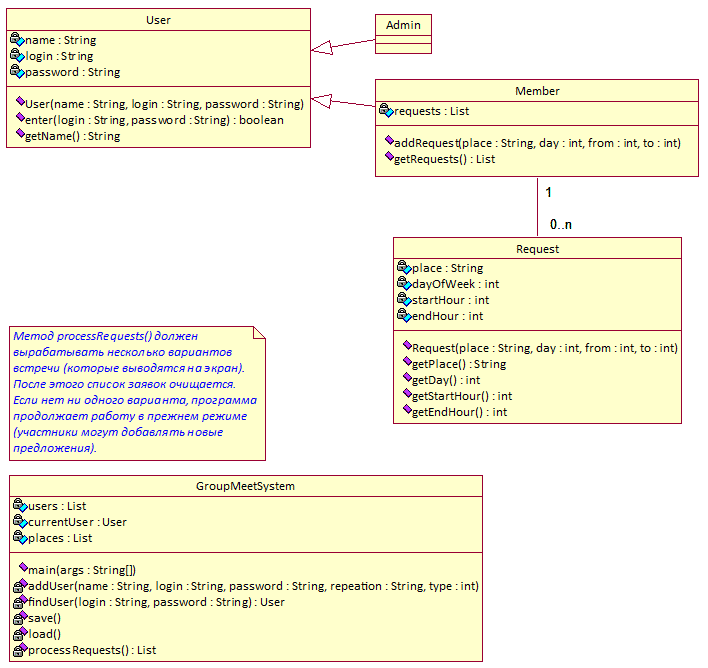
Необходимо разработать программу для координации группы участников проекта. Участникам проекта необходимо назначить время и место встречи. Каждый участник регистрируется в системе и указывает свои возможности (список промежутков времени и мест, где он может встретиться). Система подводит итоги, определяя возможные варианты времени и места встречи, подходящие для всех участников.

Возможное время встречи должно задаваться сочетанием день недели + с какого часа + до какого часа. Место встречи выбирается из списка, который заполняется заранее.

Основные алгоритмы

*Определение вариантов места встречи.*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей. Интерфейс для указания всех возможных мест встречи.
* Интерфейс для добавления возможностей.
* Интерфейс для расчета результатов.

Таблицы данных:

* Пользователи
* Места встречи
* Возможности

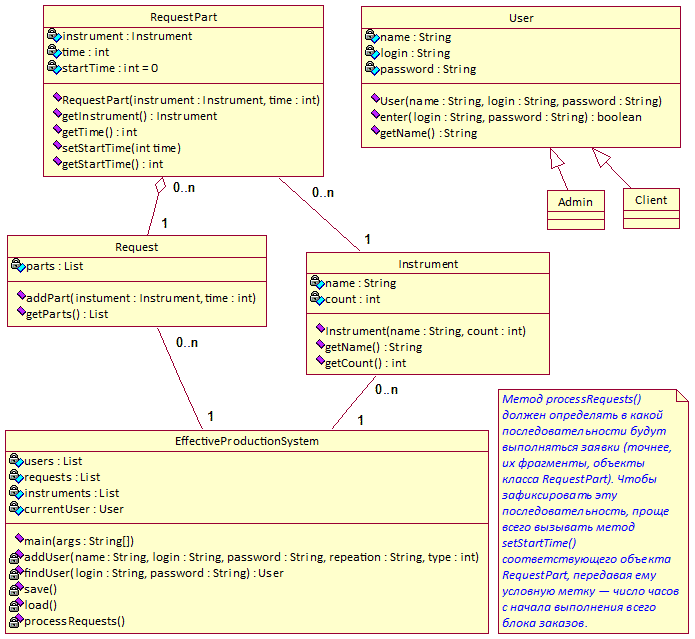
# Система оптимального распределения производственных мощностей

Написать программу для оптимального распределения производственных мощностей. В производственный отдел поступают заказы на изготовление продукции. Для выполнения каждого заказа требуется определенное количество ресурсов. Пусть в качестве ресурсов выступает время использования оборудования. Для простоты предположим, что список необходимого оборудования и времени содержится в самом заказе, причем список упорядочен в соответствии с технологическими операциями заказа (т.е. имеет значение, в каком порядке это оборудование будет предоставлено). Программа должна разработать очередность предоставления оборудования, с целью минимизировать общее время выполнения всей партии заказов.

Основные алгоритмы

*Разработка очередности предоставления оборудования*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс для указания перечня оборудования.
* Интерфейс для приема заказов.
* Интерфейс для составления плана загрузки оборудования.

Таблицы данных:

* Пользователи
* Заказы
* Оборудование

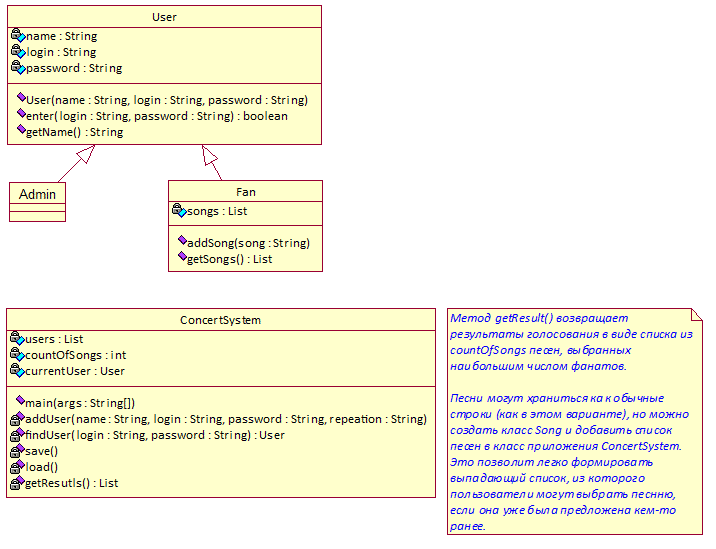
# Концерт по заявкам

Разработать систему для формирования программы концерта по заявкам. Пользователи (фанаты) регистрируются в системе и выбирают песни из предложенного списка (или добавляют свои). Каждый пользователь может выбрать любое количество песен. Когда время подачи заявок оканчивается, формируется программа концерта, включающая песни, набравшие наибольшее число заявок (количество песен определяется заранее).

Основные алгоритмы

*Составление программы концерта по заявкам*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для входа в систему.
* Интерфейс для регистрации пользователей.
* Интерфейс администратора (указание основных параметров концерта: название, цена билетов, место проведения, из которых нас интересует только число песен).
* Интерфейс для добавления песни в список
* Интерфейс для подачи заявок.
* Интерфейс для вывода результатов (программа концерта)

Таблицы данных:

* Пользователи
* Голоса

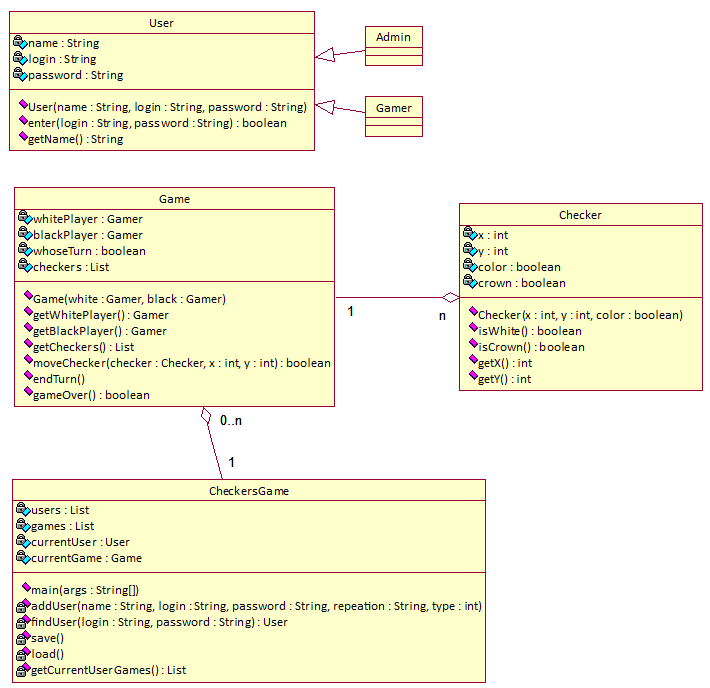
# Игра «шашки»

Написать программу для игры в шашки. Пользователи регистрируются в системе и ходят по очереди. Текущая партия может быть приостановлена, а затем продолжена заново. В продвинутой версии системы может поддерживаться одновременно несколько партий.

Основные алгоритмы

*Проверка допустимости сделанного хода. Проверка окончания игры*

Рекомендуемая диаграмма классов



Интерфейсы:

* Интерфейс для регистрации игроков.
* Интерфейс входа в систему.
* Интерфейс для начала и загрузки (выбора) партии. Интерфейс для очередного хода.
* Интерфейс для объявления результатов партии.

Таблицы данных:

* Игроки.
* Партии.
* Позиции шашек.