Факультет компьютерных технологий и прикладной математики
Кафедра вычислительных технологий
02.03.02

Паттерны программирования
Лабораторная работа № 3. Модель из одной сущности

Каждое задание должно быть загружено на личный git-репозиторий отдельным коммитом. Лабораторная работа выполняется в одной папке. Защита работы возможна на любой лабораторной работе от 1 до 16. Каждое из 4 заданий проверяется отдельно с учетом вопросов преподавателя. Задание засчитывается отдельно, лабораторная работа зачтена в случае выполения всех 4 заданий.

Если часть задач выполнена в один коммит, работа не проверяется. Если все коммиты сделаны в один час, работа не проверяется.

Задание 1 Подключение БД к модели

Задачи

1. Создать подключение к MySQL.
2. Создать таблицу student. Для этого создать отдельную директорию, в которой будут храниться файлы, реализующие изменения в структуре БД. Создать два типа файлов, в одном – изменения в структуре, в другом – автоматические скрипты заполнения данными. Сохранить выполненные Вами скрипты в гит репозиторий.
3. Создать ER модель таблицы средствами ERWin, поместить в директорию с БД.
4. Заполнить таблицу данными.
5. Протестируйте выполнение select запроса из этой таблицы из программы на ruby.
6. Если не было сделано в предыдущем задании, то модифицируйте конструкторы или создайте альтернативные конструкторы для класса Student так, чтобы они принимали в качестве аргумента хеш, протестируйте написанные изменения.
7. Если необходимо, внесите изменения в уже существующие классы, чтобы весь написанный функционал продолжал работать.
8. Реализовать класс Students_list_DB, который будет работать с созданной БД и таблицей, содержащей списки студентов. Для этого необходимо обеспечить выполнение следующих функций:
    a. Получить объект класса Student по ID
    b. get_k_n_student_short_list Получить список k по счету n объектов класса Student_short (например, вторые 20 элементов, чтобы в дальнейшем можно было листать длинный список), результат вернуть в формате Data_list.

c. Добавить объект класса Student в список (при добавлении сформировать новый ID).
d. Заменить элемент списка по ID.
e. Удалить элемент списка по ID.
f. Получить количество элементов

9. Методы работы с БД вынести в отдельный класс.
10. Привести тривиальный пример паттерна Одиночка, показать демонстрацию корректной работы данного паттерна.
11. Реализовать паттерн одиночка для класса работы с БД.
12. Отразить изменения в классах в диаграмме классов.

Вопросы.

a. Какой класс отвечает за работу с MySQL? Как осуществляется работа?

b. В каком формате объект указанного Вами класса возвращает результат SQL запроса Select?

c. Опишите структуру и основные принципы работы с коллекцией ХЕШ.

d. Опишите структуру и основные принципы работы с коллекцией множество.

e. Опишите проблему и место паттерна одиночка в разработке.

f. Напишите на бумаге пример реализации данного паттерна и способ проверки корректности его работы.

g. Попытайтесь снова сформулировать ответ на вопрос – В чем необходимость использования полей и методов класса?

h. Что такое метод класса, в чем его отличие от метода объекта? Приведите два практических примера, когда введение метода класса вы считаете необходимым согласно концепциям ООП.

Задание 2. Сериализация * (НЕ ОБЯЗАТЕЛЬНО).

Задачи.

1. * (НЕ ОБЯЗАТЕЛЬНО) Разобрать паттерн адаптер, написать тривиальный пример.
2. *** * (НЕ ОБЯЗАТЕЛЬНО) Применить паттерн адаптер к классам Students_list_DB, Students_list_txt, Students_list_JSON, Students_list_YAML, выделив класс Student_list, переведя весь функционал в стратегию, обеспечив одинаковый интерфейс с помощью паттерна адаптер.
3. * (НЕ ОБЯЗАТЕЛЬНО) Скорректировать диаграмму классов.

4. * (НЕ ОБЯЗАТЕЛЬНО) В итоге должен получиться класс Student_list, позволяющий –
    a. Получить объект класса Student по ID
    b. Получить список k по счету n объектов класса Student_short (например, вторые 20 элементов, чтобы в дальнейшем можно было листать длинный список), результат вернуть в формате Data_list.
    c. Добавить объект класса Student в список (при добавлении сформировать новый ID).
    d. Заменить элемент списка по ID.
    e. Удалить элемент списка по ID.
    f. get_student_short_count Получить количество элементов

Вопросы.

a. * (НЕ ОБЯЗАТЕЛЬНО) Опишите проблематику, в которой необходим паттерн адаптер.

b. * (НЕ ОБЯЗАТЕЛЬНО) Приведите на листе бумаге тривиальный пример реализации этого паттерна.

c. * (НЕ ОБЯЗАТЕЛЬНО) Опишите словами место в проекте, где возможно применение данного паттерна, Опишите Вашу реализацию паттерна адаптер.

d. * (НЕ ОБЯЗАТЕЛЬНО) Опишите, как вы понимаете утверждение – проектируйте на уровне интерфейсов, а не реализаций

Задание 3. Построение окна.

Задачи

1. Установить gem FXRuby.
2. Разработать приложение, содержащее три вкладки. В рамках ЛР4 и ЛР5 будет разрабатываться первая вкладка. Реализовать закрытие чёокна.
3. Первая вкладка будет отображать список студентов, Student_list_view.
4. На вкладке три области –область фильтрации, таблица, область управления(кнопки).
5. В области фильтрации установить 5 частей. 1 часть – поле для ввода фамилии и инициалов. 2 часть – содержит или три радиокнопки или комбобокс с возможностью выбора наличия или отсутствия гита, реализованного в 3 альтернативы Да, нет, Не важно. Так же во второй части поле для ввода текста (поиск по гиту). Если выбраны альтернативы Нет или Не важно, то поле недоступно, если выбрана альтернатива – Да – поле становится доступным. Подобным образом выполнены части 3, 4 и 5, отвечающие за почту, телефон и телеграмм соответственно.
6. Во второй части находится таблица, протестируйте заполнение таблицы произвольными данными (хардкод с примерами).

Обеспечьте невозможность для ПОЛЬЗОВАТЕЛЯ менять содержимое полей таблицы вручную. Реализуйте сортировку данных по полю при нажатии на заголовок этого поля в таблице на примере любого ОДНОГО поля. В области с таблицей будут находится кнопки смены страниц, в таблице отображается выбранное Вами число элементов, 20 например. Пример реализации кнопок схемы страниц остаётся на Ваш выбор, обязательное условие – ДОЛЖНО БЫТЬ ВИДНО, СКОЛЬКО ВСЕГО СТРАНИЦ.

7. В третьей области расположите кнопки – добавить, изменить, удалить, обновить. Кнопки добавить и обновить доступны всегда, кнопки изменить и удалить доступны ТОЛЬКО при выделении строки в таблице. Если выделена одна строка, доступны обе кнопки, если выделено больше одной строки доступна ТОЛЬКО кнопка удалить. Кнопка обновить должна осуществлять поиск согласно установленным фильтрам.
8. Если есть желание, добавьте дополнительные кнопки для реализации описанной выше логики, например кнопку Сбросить для сброса настроек фильтров, или текст вместо таблицы или в самой таблице, если элементов не найдено, логику изменения размера окна в случае необходимости, разное количество элементов, отображаемое в таблице и т.п.).
9. Если есть желание, разберите пример любого другого gem для построения GUI оконных приложений, например Qt, Shoes и тд

Вопросы.

i. Какой класс позволяет выстраивать окно в выбранном Вами gem?

j. Как происходит обмен сообщениями между классами, реализующими элементы графического интерфейса?

k. Как и где указываются методы обработчиков событий, как реализуются эти методы в выбранной Вами библиотеке, опишите подробно на примерах.

Далее необходимо руководствуясь паттерном MVC построить приложение, которое отображает в таблицу информацию о студентах согласно установленным значениям фильтров. Если написан текс, он ищется как подстрока, если установлен фильтр не важно, аттрибут не учитывается в поиске, если фильтры Да или Нет, то система выдаёт элементы, удовлетворяющие условиям поиска.

Задание 4. MVC – CRUD - read.

Задачи.

1. Разобрать паттерн наблюдатель, написать тривиальный пример.
2. На основании паттерна MVC классы, отвечающие за визуализацию приложения, не реализуют никакой логики, исходя из этого

необходимо Написать класс Student_list_controller, в который будет вынесена ВСЯ логика работы. При открытии приложения система создаёт экземпляр класса контроллера и хранит ссылку на контроллер в классе View и ссылку на View в классе контроллера. С учетом этого реализуйте конструктор контроллера и модифицируйте конструктор View. При загрузке контроллера обязательно должен быть создан экземпляр класса Students_list. После он должен хранится в поле указанного контроллера. Отобразите изменения в диаграмме классов.

3. При открытии приложения система ДОЛЖНА заполнить фильтры по умолчанию (позже этот пункт будет разобран отдельно). Далее алгоритм отображения следующий: объект view вызывает метод refresh_data у контроллера. Контроллер вызывает метод get_k_n_student_short_list у класса Students_list, в результате получает объект рассмотренного ранее класса data_list_student_short : Data_list_student_short, это объект data_list_student_short обязательно сохраняется в отдельном поле, чтобы дальше не создавать его заново. Созданному экземпляру data_list_student_short передать ссылку на объект View. У созданного объекта data_list_student_short вызвать метод notify, для этого внести изменения в класс модели Data_list_student_short, добавив соответствующий метод. Приведенная реализация есть реализация паттерна Наблюдатель, где View наблюдатель модели. Объект data_list_student_short в методе notify должен вызвать два метода, которые должны будут быть реализованы у каждого наблюдателя: set_table_params(column_names, whole_entities_count) и set_table_data(data_table). ПРИ ЖЕЛАНИИ ЛОГИКУ ПОСТРОЕНИЯ ТАБЛИЦЫ МОЖНО ТАКЖЕ ВЫНЕСТИ В КОНТРОЛЛЕР (пока не вижу зачем, альтернативы не будет, но стратегия, как никак)))))). Первый метод используется для задания колонок таблицы и расчета изображения кнопок из задачи 6 задания 1. Второй метод для заполнения таблицы данными. После завершения выполнения метода refresh_data система отображает окно.
4. ПОСТРОИТЬ ДИАГРАММУ ПОСЛЕДОВАТЕЛЬНОСТИ, ОПИСЫВАЮЩУЮ ВЫЗОВ КАЖДОГО МЕТОДА прецедента ВКЛЮЧИТЬ ПРОГРАММУ.
5. Скорректировать диаграмму классов.
6. * (НЕ ОБЯЗАТЕЛЬНО) Протестировать ПО на отображении данных из базы и из любого сериализованного файла (например, YAML или JSON).
7. * (НЕ ОБЯЗАТЕЛЬНО) Сформировать список исключительных ситуаций, который может возникнуть при выполнении включения программы.

8. * (НЕ ОБЯЗАТЕЛЬНО) Описать обработку исключительных ситуаций в методах КОНТРОЛЛЕРА. **ОБЯЗАТЕЛЬНО ОДНУ** ситуацию – невозможно подключиться к БД при вызове метода get_k_n_student_short_list Получить список k по счету n объектов класса Student_short
9. Протестировать обновление данных при переключении вкладок (например, открыть окно, переключить на пустую вторую вкладку, залезть в файл, зачистить половину, вернуться к ПО, перейти на первую вкладку, в результате должно отображаться измененное содержимое).
10. Модифицировать для этого метод refresh_data так, чтобы НЕ МЕНЯЛСЯ экземпляр data_list_student_short, а лишь менялось его наполнение(таблица), вспомните, как мы проводили проверку на идентичность объектов. Важно, так как в дальнейшем при выполнении любой операции мы будем вызывать именно этот метод.
11. Обеспечьте смену элементов в таблице по нажатию на кнопки смены страниц из задачи 6 задания 1. Модифицировать диаграмму классов.

Вопросы.

a. Опишите проблематику, в которой необходим паттерн наблюдатель.

b. MVC. Общий подход. Пример диаграммы последовательности реализации операции CRUD - read.

c. Опишите словами место в проекте, где возможно применение данного паттерна наблюдатель, Опишите Вашу реализацию паттерна адаптер. что есть стратегия в MVC?

d. Напишите на листочку схему реализации паттерна наблюдатель.