Факультет компьютерных технологий и прикладной математики
Кафедра вычислительных технологий
02.03.02

Паттерны программирования
Лабораторная работа № 2. Классы ядра и модели

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 6 заданий проверяется отдельно с учетом вопросов преподавателя. Задание засчитывается отдельно, лабораторная работа зачтена в случае выполения всех 6 заданий.

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

Часть заданий выполняется по вариантам.

Задание 1. Объекты и классы

ВАЖНО. КАЖДАЯ ЗАДАЧА – ОТДЕЛЬНЫЙ КОММИТ. При защите описывать изменения и их необходимость.

Задачи

1. Создать класс Student в отдельном файле с полями объекта ID, Фамилия, Имя, Отчество, Телефон, Телеграм, Почта, Гит. ФИО – обязательно, остальные – нет. Написать конструктор, написать геттер и сеттер для каждого поля, при именовании методов ОБЯЗАТЕЛЬНО пользоваться соглашениями о наименованиях в ruby.
2. Создать несколько объектов из отдельного файла main.rb вывести информацию о них на экран. Продумать корректный способ вывода информации о текущем состоянии объекта на экран, модифицировать класс.
3. Избежать дублирования кода в конструкторе, геттере и сеттере.
4. Воспользоваться аттрибутами для краткого написания геттеров и сеттеров.
5. Обеспечить в конструкторе возможность создания объектов с любой комбинацией заполнения необязательных полей.
6. *** Продумать аргументы конструктора в форме ХЭШа.
7. Добавить метод КЛАССА, проверяющий, является ли некоторая строка телефонным номером. Модифицировать класс так, чтобы в произвольный момент времени не мог существовать объект с непозволительной строкой в поле

номер телефона, для этого придётся модифицировать конструкторы. Протестировать полученный класс.

8. Создать валидации для корректной формы строки в остальных полях.
9. Напишите метод validate, который проводит две валидации наличие гита и наличие любого контакта для связи, по возможности разделить методы.
10. Напишите метод set_contacts, который устанавливает значения поля или полей для введенных контактов.
11. Начать построение диаграммы классов, описав данный класс.

Вопросы.

a. Что такое класс, что такое объект, как создать объект класса?

b. В чем заключается принцип инкапсуляции? Как получить доступ к полям объекта из внешнего класса?

c. Как используются символы для решения задач инкапсуляции и уменьшения количества кода при описании класса?

d. Что такое конструктор, зачем он нужен, как описывается конструктор в произвольном классе?

e. Опишите механизм создания объекта.

f. 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. Чтение, просмотр и запись сущностей

ВАЖНО. КАЖДАЯ ЗАДАЧА – ОТДЕЛЬНЫЙ КОММИТ. При защите описывать изменения и их необходимость.

1. Продумать структуру String представления объекта класса, согласовать его с пунктом 2. Написать отдельный конструктор, принимающий на вход строку, который эту строку парсит и вызывает стандартный конструктор с параметрами. Протестировать конструктор из класса main.
2. Продумать структуру исключений для данного конструктора в случае, если парсинг строки невозможен, и в случае, если данные в строке не прошли валидацию.
3. Напишите метод getInfo, который возвращает краткую информацию о студенте – Фамилия и Инициалы; гит, связь (указать любое средство связи и указать, какое оно) в ОДНОЙ строке. По возможности, разделить метод на составляющие (в дальнейшем эта инфа будет выводится в таблице и надо будет брать её по частям).

4. Напишите класс Student_short, имеющий 4 поля ID, ФамилияИнициалы, гит, контакт. Поля нельзя редактировать. Возможно два конструктора – в одном – объект класса Student, в другом ID и строка, содержащая всю остальную информацию.
5. Провести рефакторинг, выделив суперкласс и убрав дублирование кода в классах Student и Student_short.
6. Отметить изменения в диаграмме классов.
7. Написать метод read_from_txt, который получает аргументов адрес файла, выбрасывает исключение с оповещением, если адрес некорректен, и возвращает массив объектов типа Student.
8. Протестировать указанный метод, организовав вывод краткой информации по каждому объекту.
9. Написать метод write_to_txt, который получает в качестве аргументов адрес и имя файла и список объектов типа Student.
10. Протестировать совместимость данных методов.

Вопросы

a. Опишите структуру классов языка Ruby, как в нее вписывается написанный Вами класс?

b. Опишите принцип наследования, переопределение методов и способы вызова переопределенного метода.

c. Опишите принципы работы конструкторов для наследуемых классов.

d. Какие методы объекта обязательно есть у любого написанного Вами класса, опишите, что они делают.

Задание 3. Классы модели.

Задачи

1. Создать класс Data_table. Данный класс имеет поле – двумерный массив с элементами любого типа, поле недоступно ни для чтения, ни для записи. Строка в таблице – значения атрибутов сущности. Запись данных только с помощью конструктора.
2. Реализовать методы, позволяющие – получить элемент по номеру строки и столбца БЕЗ возможности его редактировать, получить количество столбцов в таблице, получить количество строк в таблице.
3. Создать класс Data_list, содержащий упорядоченный массив элементов любого класса. Любые изменения данных

недоступны, кроме одного метода который, будет обговорен будет позже.

4. Данный класс должен реализовывать следующий методы:
    a. select(number) Выделить элемент по номеру
    b. get_selected Получить массив id выделенных элементов
    c. get_names Получить массив наименований аттрибутов, кроме ID (0 столбцец - № по порядку). данный метод не будет работать для указанного класса, так как не имеет информации об объектах внутри, реализовать этот класс необходимо в наследниках.
    d. get_data: Получить объект класса Data_table, где нулевой столбец – сгенерированный № по порядку, остальные столбцы заполнены ВСЕМИ атрибутами сущности, кроме ID, данный метод не будет работать для указанного класса, так как не имеет информации об объектах внутри, реализовать этот класс необходимо в наследниках.

5. Создать наследника Data_list_student_short, реализующего методы get_data и get_names для класса Student_short. Использовать методы родителя в переопределенном методе.

6. Разобрать на простейшем примере паттерн Шаблон(Паттерн Паттерн).

7. Использовать данный паттерн в методах get_data и get_names, реализовать дополнительные private методы в наследнике.

8. Протестировать на примере формирования сущностей класса Student_short. Важно, конструктор должен быть независимым от класса.

9. Реализовать сеттер для массива объектов, заменяющий массив объектов, при этом массив наименований столбцов НЕ МЕНЯЕТСЯ никогда, модифицировать конструкторы.

10. Отметить указанные классы в диаграмме классов

Вопросы

a. Опишите принцип инкапсуляции на примере написанных классов, объясните наличие или отсутствие геттеров и сеттеров для каждого из полей разработанных Вами классов.

b. Опишите, в каком случае необходимо использовать паттерн Шаблон. Опишите его преимущества и недостатки.

c. В чем заключается принцип подстановки?

d. Напишите на листочке пример рефакторинга на основании паттерна Шаблон.

Задание 4. Классы Students_list_txt, Students_list_JSON, Students_list_YAML

Задачи

1. Реализовать класс Students_list_txt, который будет работать с текстовым файлом, содержащим списки студентов. Воспользоваться уже реализованными методами, перенеся их функционал в данный класс. Для этого необходимо обеспечить выполнение следующих функций:
    a. Чтение всех значений из файла;
    b. Запись всех значений в файл;
    c. Получить объект класса Student по ID
    d. 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. Учесть в виде необязательного аргумента существующий объект класса data_list, вернуть существующий или измененный объект в зависимости от принятого аргумента. Провести проверку получаемого объекта отдельно из файла main (Провести рефакторинг, убрав переопределение метода get_names из наследников Data_list, объяснить, в чем выигрыш.)
    e. Сортировать элементы по набору ФамилияИнициалы.
    f. Добавить объект класса Student в список (при добавлении сформировать новый ID).
    g. Заменить элемент списка по ID.
    h. Удалить элемент списка по ID.
    i. get_student_short_count Получить количество элементов

2. Реализовать класс Students_list_JSON, который будет работать с текстовым файлом, содержащим списки студентов. Для этого необходимо обеспечить выполнение следующих функций:
    a. Чтение всех значений из файла;
    b. Запись всех значений в файл;
    c. Получить объект класса Student по ID
    d. 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. Учесть в виде необязательного аргумента существующий объект класса data_list, вернуть или существующий, или измененный объект.
    e. Сортировать элементы по набору ФамилияИнициалы.

f. Добавить объект класса Student в список (при добавлении сформировать новый ID).
g. Заменить элемент списка по ID.
h. Удалить элемент списка по ID.
i. get_student_short_count Получить количество элементов

3. Реализовать класс Students_list_YAML, который будет работать с текстовым файлом, содержащим списки студентов. Для этого необходимо обеспечить выполнение следующих функций:
   a. Чтение всех значений из файла;
   b. Запись всех значений в файл;
   c. Получить объект класса Student по ID
   d. get_k_n_student_short_list Получить список k по счету n объектов класса Student_short (например, вторые 20 элементов, чтобы дальше можно было листать длинный список), результат вернуть в формате Data_list.
   e. Сортировать элементы по набору ФамилияИнициалы.
   f. Добавить объект класса Student в список (при добавлении сформировать новый ID).
   g. Заменить элемент списка по ID.
   h. Удалить элемент списка по ID.
   i. get_student_short_count Получить количество элементов

4. Провести рефакторинг, создав супер класс и выделив повторяющийся код в его методы. Модифицировать диаграмму классов.

5. Написать тривиальный пример паттерна СТРАТЕГИЯ.
6. Провести рефакторинг, применив паттерн стратегия для указанных трех классов. Модифицировать диаграмму классов.

Вопросы

a. Опишите, как Вы понимаете утверждение – Используйте делегацию вместо наследования, напишите на бумаге тривиальный пример.
b. Что такое отношение ассоциации в ОПП? Покажите разновидности ассоциации.
c. Опишите проблему и место применения паттерна стратегия, как пример делегации.
d. Напишите на бумаге или на доске тривиальный пример наследования и реализации паттерна стратегия, объясните разницу.
e. Приведите пример ситуации, когда нет необходимости в применении паттерна.