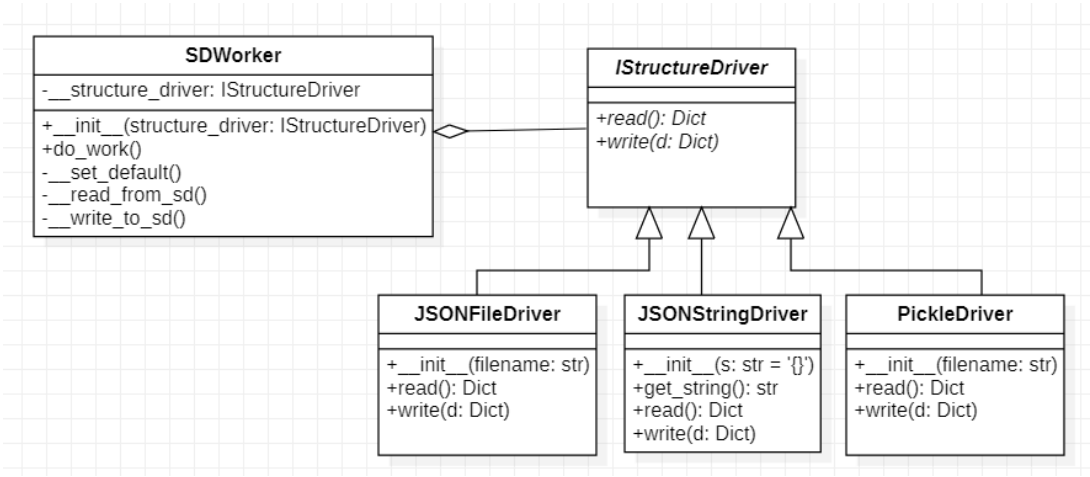
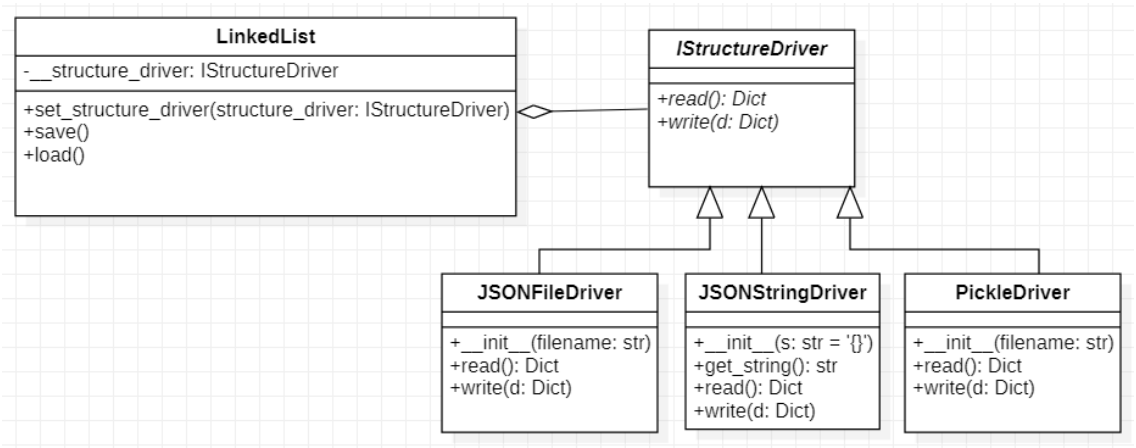
Сериализация (сохранение) и десериализация (загрузка) двусвязного списка в файл (Шаблон проектирования «Стратегия»)

В лабораторной работе № 1 последнее задание – разработка двунаправленного (двусвязного) линейного списка (ДЛС). Теперь добавим возможность сохранять и загружать этот список из внешнего источника. Будем использовать файл. На лекции по шаблонам проектирования был разработан драйвер для сохранения данных в форматы JSON и Pickle. Для универсализации выбора формата сохранения и загрузки был использован шаблон проектирования «Стратегия».

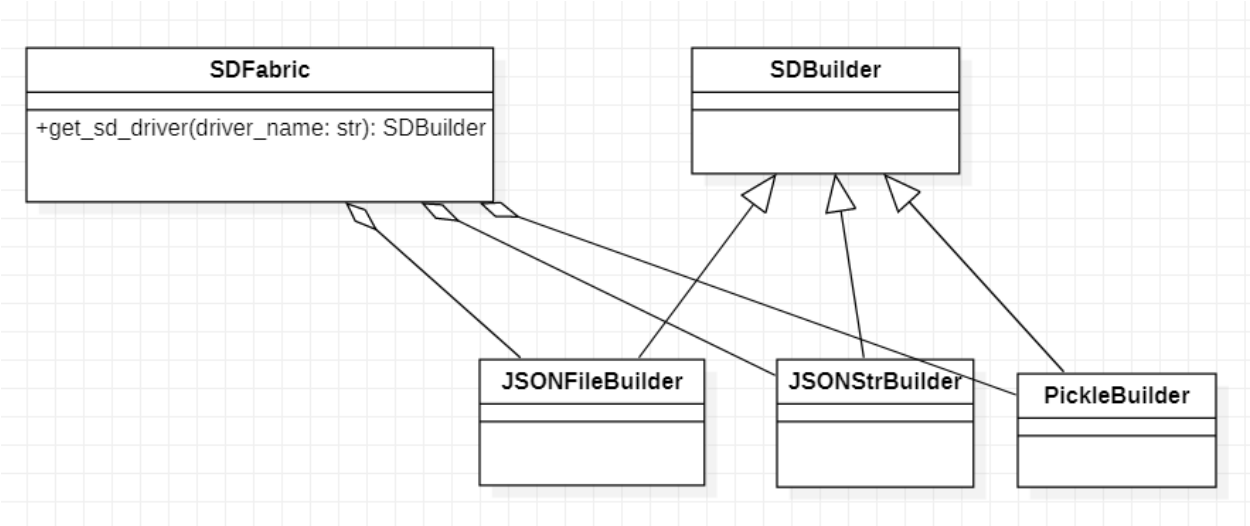


В качестве входного параметра используется словарь. Для упрощения работы с алгоритмами сохранения и чтения данных необходимо разработать некоторый объект, который будет себя вести как словарь. Данный подход применим в следующей лабораторной работе (шаблон проектирования «Заместитель»), потому что для использования данного шаблона для LinkedList необходимо уметь перегружать встроенные операторы. Сейчас поступим следующим образом. Добавьте методы LinkedList.save и LinkedList.load. В этих методах сделайте преобразование из LinkedList в словарь. В результате будет следующая диаграмма классов. Код IStructureDriver и всех наследников не меняется.



Выбор драйвера через input
(Шаблоны проектирования «Строитель» и «Фабрика»)

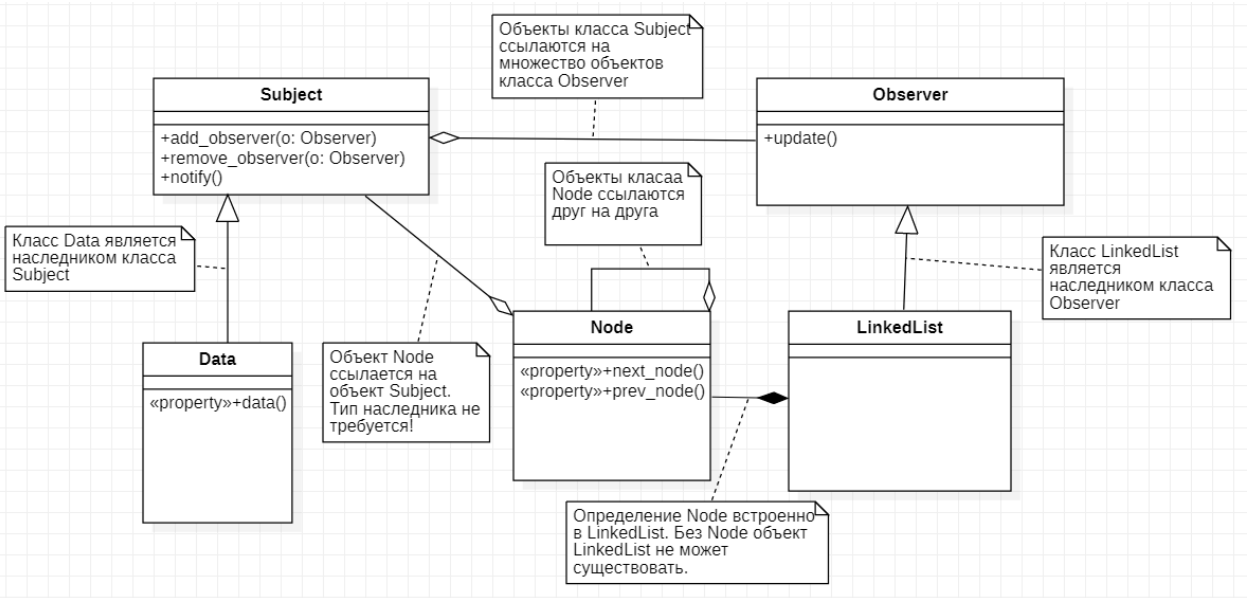
Встройте код из лекции для выбора драйвера для LinkedList.




```
driver_name = input('Please enter driver name >')

builder = SDFabric().get_sd_driver(driver_name)
sd = builder.build()

ll = LinkedList()

ll.set_structure_driver(sd)
```

Добавление автоматического сохранения в файл при изменении данных в любом элементе списка, добавлении или удалении элемента списка (реактивное программирование)

Как будет работать код. Если мы изменяем данные в Node, то должен вызваться список, чтобы сохранить его в файл. Для реализации данного подхода необходимо, чтобы данные, которые добавлялись список были наследниками Subject, чтобы они могли оповещать список о своём изменении. А класс LinkedList был наследником Observer, чтобы он мог оповещаемым при изменении данных в Node.



Класс Data – это обёртка под простые типы данных, которые используются в Python. При изменении данных в объектах мы не можем никак оповестить сам объект что он изменился. Для этого нам нужна эта обёртка. Реализуется она очень просто. Создайте приватную переменную Data.__data и свойство для чтения и для записи данных. При записи данных в свойстве set просто вызовите метод notify. В этот момент список будет оповещён о том, что какие-то данные были изменены в списке. Код классов Subject и Observer не меняется.

И последний шаг. Измените метод класса LinkedList.insert. Примерная реализация:

```
class Data(Subject):
    pass

class LinkedList(Observer):


    def insert(self, data: Subject):
        if not isinstance(data, Subject):
            raise TypeError("Data must be inheritance from Subject")
        data.add_observer(self) # Добавляем себя в качестве слушателя

        # Вставляем в список
```

Избавление от циклической зависимости

Изучите как ссылаются объекты в данной работе. Образуется циклическая зависимость. Замените класс Subject на WeakSubject, в котором реализуйте слабые ссылки.

Подсказка: требуется изменить три строки.