**Лабораторная работа 1. 3 часть**

**Приложение «Телефонный справочник».**

**Использование паттерна Стратегия для сохранения и открытия.**

**Задание**. Реализуем в приложении «Телефонный справочник» с помощью шаблона проектирования Strategy (рис. 1) три следующих ситуации:

1. сохранение данных в файл,
2. открытие данных из файла, при этом предыдущие данные стираются (предварительно нужно их сохранить),
3. добавление данных из фала, при этом предыдущие данные остаются, а из файла добавляются только новые данные, которых в исходном списке нет.

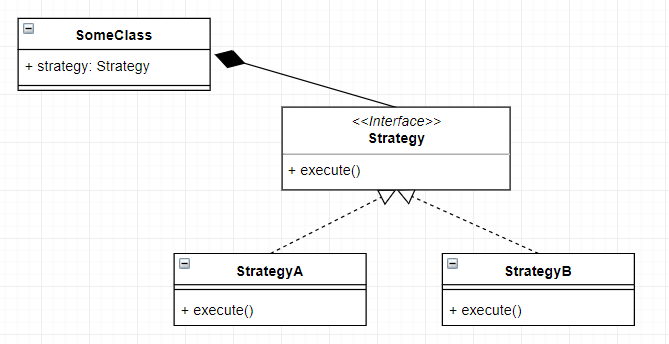


Рисунок 1. UML-схема паттерна Strategy.

Создайте в проекте папку Save. В ней будет храниться все, что касается сохранения данных. Добавьте в папку интерфейс с именем ISave. В интерфейсе опишите сигнатуру метода сохранения. Метод должен принимать два параметра: ЧТО сохранить – список данных, КУДА сохранить – имя файла, в которые данные сохраняются.

namespace Название\_пространства\_имен\_проекта.Save

{

interface ISave

{

void Save(List<Note> notes, String fileName);

}

}

В проекте уже реализованы два способа сохранения данных: в txt-файл и в xml-файл. Вынесем эту логику из класса формы в специальные классы. Опишите два класса в папке Save: SaveTxt и SaveXml.

Например, реализация класса SaveTxt может быть такой:

class SaveTxt : ISave

{

public void Save(List<Note> notes, string fileName)

{

using (StreamWriter sw = new StreamWriter(fileName))

{

foreach (Note MyRecord in notes)

{

sw.WriteLine(MyRecord.LastName);

sw.WriteLine(MyRecord.Name);

sw.WriteLine(MyRecord.Patronymic);

sw.WriteLine(MyRecord.Street);

sw.WriteLine(MyRecord.House);

sw.WriteLine(MyRecord.Flat);

sw.WriteLine(MyRecord.Phone);

}

}

}

}

Реализуйте класс SaveXml самостоятельно.

Сейчас нужно использовать данные классы в программе. Опишите в классе MainForm поле:

ISave saveBehavior;

В событии нажатия на кнопку сохранения вызовите необходимый конструктор. Например, если в Фильтре диалога сохранения выбран формат файла txt, то вызовите конструктор сохранения в файл txt:

saveBehavior = new SaveTxt();

Для xml файла нужно вызвать свой конструктор. Так как переменная saveBehavior изначально имеет тип интерфейса ISave, то есть в качестве базового класса здесь указывается интерфейс.

После создания экземпляра класса нужно просто вызвать метод save():

saveBehavior.Save(PhoneNote,path);

Таким образом вся реализацию была убрана в соответствующий класс, при этом вы точно знаете, какой из классов для чего применяется. Класс формы «разгрузили» от громоздкого кода.

**Запуск**. Запустите и убедитесь, что программа работает исправно.

Аналогичным образом нужно поступить с открытием файла. Создать папку Open, добавить в нее интерфейс IOpen и два класса OpenTxt и OpenXml.

Интерфейс IOpen должен содержать два метода: для создания нового списка из файла и для добавления новых данных из файла.

Реализуйте это поведение самостоятельно.

**Запуск**. Запустите и убедитесь, что функции сохранения данных, открытия и добавления данных из файла работают исправно.

**ВОПРОС**. Зачем создавать два интерфейса отдельно для сохранения и для открытия, а не один интерфейс для всего. Какой и принципов SOLID нужно использовать, чтобы ответить на этот вопрос. Какие еще принципы SOLID задействованы в этом решении.