МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Е.Н. ЭГОВ

РАЗРАБОТКА КОМПОНЕНТО-ОРИЕНТОВАННЫХ ПРОГРАММ

практикум по дисциплине

«Компоненто-ориентированное программирование»

Ульяновск

УлГТУ

2018

УДК 004.9 : 338(076)

ББК 32.973.26 – 018.2я73

К69 Рекомендовано научно-методической комиссией факультета информационных систем и технологий в качестве практикума.

Эгов, Евгений Николевич

РАЗРАБОТКА КОМПОНЕНТО-ОРИЕНТОВАННЫХ ПРОГРАММ : практикум / Е. Н. Эгов. – Ульяновск : УлГТУ, 2018. – 188 с.

К69

Практикум адресован студентам для выполнения и оформления лабораторных работ по дисциплине «Компоненто-ориентированное программирование». Рекомендации и требования к лабораторным работам разработаны в соответствии с рабочей программой дисциплины. Предназначены для студентов, обучающихся по направлению 09.03.04 «Программная инженерия».

Работа подготовлена на кафедре «Информационные системы».

УДК 004.9 : 338(076)

ББК 32.973.26 – 018.2я73

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1.
СОЗДАНИЕ ВИЗУАЛЬНОГО КОМПОНЕНТА

Цель

Научится создавать визуальные компоненты и встраивать их в другие проекты.

Визуальные компоненты применяются с целью уменьшения количества кода (если часть логики идет в компоненте и потом используется в других компонентах или формах, например, вывод списка значений перечисления в выпадающий список с возможностью получения выбранного значения и использование этого выпадающего списка в нескольких формах) или для разработки сложных компонент (состоит из нескольких взаимосвязанных компонент, например, DataGridView с кнопками для вызова форм добавления, редактирования и удаления записей). Также, компоненты, с расширенным функционалом и дизайном по сравнению со стандартными компонентами, включены в наборы платных библиотек (например, devexpress). Разработка компонент является как бы более глубоким уровнем программирования, так как пользователями данной продукцией будет не конечные пользователи, как при разработке, например, коммерческих приложений, а программисты, которые будут использовать эти компоненты в своих приложения.

Задание

В рамках первой лабораторной вам предстоит выполнить следующие задачи:

1. *Создать библиотеку, в которой реализовать визуальные компоненты (3 штуки) по заданию.*
2. *Создать описание компонентов, какие есть методы, события, свойства у компонент и как их использовать (будет включено в итоговый отчет).*
3. *Реализовать десктопное приложение (Windows Forms), для проверки работоспособности компонент.*

Все компоненты разбиты условно на 3 блока (выбора, ввода значения и вывода списка). В каждом блоке представлено по 3 различных варианта используемых компонент или 3 варианта работы с компонентом. Также для некоторых компонент имеются 3 варианта загрузки данных, необходимых для их работы.

Блоки визуальных компонент:

- **Визуальный компонент выбора:**
  - Визуальный компонент для выбора из выпадающего списка (ComboBox). Список заполняется строковыми значениями. Заполнение может происходит 3 возможными способами, в зависимости от варианта:
    - через метод, у которого в передаваемых параметрах идет список строк и через этот список идет заполнение ComboBox;
    - через метод, у которого в передаваемых параметрах идет строка, которая добавляется в Items элемента ComboBox;
    - через публичное свойство, которое передает ссылку на свойство Items элемента ComboBox, через которое и идет заполнение.

    Отдельный публичный метод отчистки списка. Публичное свойство (set, get) для установки и получения выбранного значения (возвращает пустую строку, если нет выбранного значения), а также событие, вызываемое при смене значения в ComboBox.
  - Визуальный компонент для выбора из списка с выбором (CheckedListBox). Список заполняется строковыми значениями. Заполнение может происходит 3 возможными способами, в зависимости от варианта:

- через метод, у которого в передаваемых параметрах идет список строк и через этот список идет заполнение CheckedListBox;
- через метод, у которого в передаваемых параметрах идет строка, которая добавляется в Items элемента CheckedListBox;
- через публичное свойство, которое передает ссылку на свойство Items элемента ComboBox, через которое и идет заполнение.

Отдельный публичный метод отчистки списка. Публичное свойство (set, get) для установки и получения выбранного значения (возвращает пустую строку, если нет выбранного значения), а также событие, вызываемое при смене значения в CheckedListBox.

- Визуальный компонент для выбора из обычного списка (ListBox). Список заполняется строковыми значениями. Заполнение может происходит 3 возможными способами, в зависимости от варианта:
  - через метод, у которого в передаваемых параметрах идет список строк и через этот список идет заполнение ListBox;
  - через метод, у которого в передаваемых параметрах идет строка, которая добавляется в Items элемента ListBox;
  - через публичное свойство, которое передает ссылку на свойство Items элемента ListBox, через которое и идет заполнение.

  Отдельный публичный метод отчистки списка. Публичное свойство (set, get) для установки и получения выбранного значения (возвращает пустую строку, если нет выбранного значения), а также событие, вызываемое при смене значения в ListBox.

- **Визуальный компонент ввода:**
  - Текстовый визуальный компонент для ввода значения определенного типа (TextBox), предусматривающий возможность пустого (не заполненного значения). В контроле должен быть CheckBox, который при проставленной галочке будет обозначать признак незаполненного значения (null) и неактивным элементом ввода значения (TextBox). При не проставленной галочке элемент ввода значения должен быть активным для заполнения.

    Также требуется свойство для установки и получения введенного значения (set, get) (предусмотреть возможность возврата значения null). Если у CheckBox не проставлена галочка, а в элементе не введено значение, выдавать ошибку. Если введенное значение не соответствует требуемому типу, выдавать ошибку.
  - Текстовый визуальный компонент для ввода значения с проверкой по шаблону (TextBox). Шаблон, по которому будет проверяться вводимое значение, должен заполняться через публичное свойство. Должна всплывать подсказка (например, ToolTip) с примером правильного ввода (пример должен заполняться через публичный метод). Публичное свойство для установки и получения введенного значения (set, get). При получении должна проводиться проверка, если введенное значение не соответствует шаблону, выдавать ошибку.
  - Визуальный компонент для ввода значения с проверкой вхождения в диапазон (TextBox, DateTimePicker, NumericUpDown). Должно быть 2 публичных поля для настройки границ диапазона. Публичное свойство для установки и получения введенного значения (set, get). При получении должна проводиться проверка, если введенное значение не входит в диапазон, выдавать ошибку.

- **Визуальный компонент вывода списков:**
  - Визуальный компонент для вывода списка в виде таблицы (DataGridView). DataGridView настроен так, чтобы выбиралась только одна строка и строка целиком, а не ячейка, RowHeaders был скрыт. Отдельный метод для конфигурации столбцов. Через метод указывается сколько колонок в DataGridView добавлять, их заголовки, ширину, признак видимости и имя свойства/поля объекта класса, записи которого будут в таблице выводиться, значение из которого потребуется выводить в ячейке этой колонки. Метод отчистки строк. Публичное свойство для установки и получения индекса выбранной строки (set, get). Публичный параметризованный метод для получения объекта из выбранной строки (создать объект и через рефлексию заполнить свойства его).

    Заполнение может происходит 3 возможными способами, в зависимости от варианта:

    - через **параметризованный** метод, у которого в передаваемых параметрах идет список объектов какого-то класса и через этот список идет заполнение DataGridView;
    - через **параметризованный** метод, у которого в передаваемых параметрах идет объект какого-то класса, который добавляется в новую строку DataGridView (в конец таблицы);
    - через **параметризованный** метод, у которого в передаваемых параметрах идет объект какого-то класса, номер строки и номер столбца и требуется заполнить значением из определенного свойства/поля объекта (имя свойства/поля должно быть в конфигурации столбца) ячейку на пересечении строки и столбца (если строк меньше, чем указано, то добавить строки).

- Визуальный компонент для вывода списка в виде списка записей (ListBox). Строки ListBox заполняются на основе макетной строки. В макетной строке указывается некая фраза (например, «Температура воздуха {T}, давление {P}»). Вместо {T} и {P} подставляются значения из свойств/полей объекта какого-то класса. Из каких именно свойств/полей вытаскиваются значения, указывается внутри фигурных скобок (в примере это будет имена свойств/полей T и P). Также задаются символы, которые будут означать начало и конец имени свойства в макетной строке (в примере это будут символы ′{′ и ′}′). Макетная строка и символы начала и конца для имени свойства заполняется через публичный метод. Публичное свойство для установки и получения индекса выбранной строки (set, get). Публичный параметризованный метод для получения объекта из выбранной строки (создать объект и через рефлексию заполнить свойства его).

  Заполнение может происходит 3 возможными способами, в зависимости от варианта:

  - через **параметризованный** метод, у которого в передаваемых параметрах идет список объектов какого-то класса и через этот список идет заполнение ListBox;
  - через **параметризованный** метод, у которого в передаваемых параметрах идет объект какого-то класса, который добавляется в новую строку ListBox (в конец списка);
  - через **параметризованный** метод, у которого в передаваемых параметрах идет объект какого-то класса, номер строки и имя свойства/поля, значение которого требуется подставить в указанной строке, если этого свойство/поле там еще не заполнено (если строк меньше, чем указано, то добавить строки).

- Визуальный компонент для вывода списка в виде дерева (TreeView). Для дерева задается иерархия (последовательность) из имен свойств/полей какого-то класса. Заполнение дерева происходит по следующему принципу: для объекта класса, который добавляется в дерево, вытаскивается значение (в виде строки) из свойства/поля, которое идет первым в иерархии. Если в корне дерева уже есть ветка с таким названием, то идет переход в нее (если для этого свойства/поля не указано создавать всегда новую ветку), иначе происходит создание новой ветки. Далее вытаскивается значение из второго (указано вторым в иерархии) свойства/поля добавляемого объекта и в ветке ищется подветка с таким названием. Если такая есть и для нее не указано всегда создавать новую ветку, то идет переход в нее, иначе создается новая. И т.д. по всем значениям из иерархии. Иерархия задается через публичный метод. Например, имеется класс «Сотрудник», у которого имеются поля «ФИО», «Должность» и «Отдел» (могут быть и иные поля, но нас будут интересовать эти). И требуется построить дерево так, чтобы все сотрудники были сгруппированы по отделам и должностям. В таком случае, в дереве заводим иерархию «Отдел», «Должность», «ФИО» и заносим в такое дерево список имеющихся сотрудников так, чтобы для поля «ФИО» всегда создавались новые ветки (на случай, если у нас есть полные однофамильцы). Получается на первом уровне будут указаны отделы, в них должности и в них ФИО сотрудников. Публичное свойство для установки и получения индекса выбранной ветки (set, get). Публичный метод для получения выбранной записи (для конечного элемента дерева).

  Заполнение может происходит 3 возможными способами, в зависимости от варианта:

- через **параметризованный** метод, у которого в передаваемых параметрах идет список объектов какого-то класса и через этот список идет заполнение TreeView;
- через **параметризованный** метод, у которого в передаваемых параметрах идет объект какого-то класса, который добавляется в TreeView (новая ветка или подветка, если значения свойств уже пересекаются в дереве);
- через **параметризованный** метод, у которого в передаваемых параметрах идет объект какого-то класса, и имя свойства/поля, до которого согласно иерархии, следует сформировать ветки.

Каждому студенту из группы назначется по одному компоненту из каждого из блоков (см. таблицу с вариантми). Получается, каждому нужно будет реализовать 3 компонента: один компонент выбора, один компонент ввода и один компонент вывода списка.

Варианты

ОПИСАНИЕ ФУНКЦИОНАЛА СМОТРИТЕ В ПУНКТЕ «БЛОКИ ВИЗУАЛЬНЫХ КОМПОНЕНТ»

№ вар.	Визуальный компонент выбора	Визуальный компонент ввода	Визуальный компонент вывода списков
1.	Выпадающий список. Список заполняется через метод, передающий список строк	Поле для ввода адреса электронной почты (адрес электронной почты должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий список объектов
2.	Список с выбором. Список заполняется через метод, передающий список строк	Поле для заполнения даты с проверкой, что дата находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий список объектов

№ вар.	Визуальный компонент выбора	Визуальный компонент ввода	Визуальный компонент вывода списков
3.	Список значений. Список заполняется через метод, передающий список строк	Поле для ввода целочисленного значения с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий список объектов
4.	Выпадающий список. Список заполняется через метод, передающий строку	Поле для ввода даты (формат даты должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий объект
5.	Список с выбором. Список заполняется через метод, передающий строку	Поле для заполнения текста с проверкой, что длина введенного текста находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий объект
6.	Список значений. Список заполняется через метод, передающий строку	Поле для ввода вещественного значения с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий объект
7.	Выпадающий список. Список заполняется через публичное свойство	Поле для ввода номера телефона (номер телефона должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий объект, номер строки и имя свойства/поля
8.	Список с выбором. Список заполняется через публичное свойство	Поле для заполнения числа с проверкой, что число находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий объект и имя свойства/поля
9.	Список значений. Список заполняется через публичное свойство	Поле для ввода даты (формат DD.MM.YYYY) с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий объект, номер строки и номер столбца

№ вар.	Визуальный компонент выбора	Визуальный компонент ввода	Визуальный компонент вывода списков
10.	Выпадающий список. Список заполняется через метод, передающий список строк	Поле для ввода адреса электронной почты (адрес электронной почты должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий список объектов
11.	Список с выбором. Список заполняется через метод, передающий список строк	Поле для заполнения даты с проверкой, что дата находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий список объектов
12.	Список значений. Список заполняется через метод, передающий список строк	Поле для ввода целочисленного значения с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий список объектов
13.	Выпадающий список. Список заполняется через метод, передающий строку	Поле для ввода даты (формат даты должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий объект
14.	Список с выбором. Список заполняется через метод, передающий строку	Поле для заполнения текста с проверкой, что длина введенного текста находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий объект
15.	Список значений. Список заполняется через метод, передающий строку	Поле для ввода вещественного значения с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий объект
16.	Выпадающий список. Список заполняется через публичное свойство	Поле для ввода номера телефона (номер телефона должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий объект, номер строки и имя свойства/поля

№ вар.	Визуальный компонент выбора	Визуальный компонент ввода	Визуальный компонент вывода списков
17.	Список с выбором. Список заполняется через публичное свойство	Поле для заполнения числа с проверкой, что число находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий объект и имя свойства/поля
18.	Список значений. Список заполняется через публичное свойство	Поле для ввода даты (формат DD.MM.YYYY) с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий объект, номер строки и номер столбца
19.	Выпадающий список. Список заполняется через метод, передающий список строк	Поле для ввода адреса электронной почты (адрес электронной почты должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий список объектов
20.	Список с выбором. Список заполняется через метод, передающий список строк	Поле для заполнения даты с проверкой, что дата находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий список объектов
21.	Список значений. Список заполняется через метод, передающий список строк	Поле для ввода целочисленного значения с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий список объектов
22.	Выпадающий список. Список заполняется через метод, передающий строку	Поле для ввода даты (формат даты должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий объект
23.	Список с выбором. Список заполняется через метод, передающий строку	Поле для заполнения текста с проверкой, что длина введенного текста находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий объект

№ вар.	Визуальный компонент выбора	Визуальный компонент ввода	Визуальный компонент вывода списков
24.	Список значений. Список заполняется через метод, передающий строку	Поле для ввода вещественного значения с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий объект
25.	Выпадающий список. Список заполняется через публичное свойство	Поле для ввода номера телефона (номер телефона должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий объект, номер строки и имя свойства/поля
26.	Список с выбором. Список заполняется через публичное свойство	Поле для заполнения числа с проверкой, что число находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий объект и имя свойства/поля
27.	Список значений. Список заполняется через публичное свойство	Поле для ввода даты (формат DD.MM.YYYY) с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий объект, номер строки и номер столбца
28.	Выпадающий список. Список заполняется через метод, передающий список строк	Поле для ввода адреса электронной почты (адрес электронной почты должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий список объектов
29.	Список с выбором. Список заполняется через метод, передающий список строк	Поле для заполнения даты с проверкой, что дата находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий список объектов
30.	Список значений. Список заполняется через метод, передающий список строк	Поле для ввода целочисленного значения с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий список объектов

№ вар.	Визуальный компонент выбора	Визуальный компонент ввода	Визуальный компонент вывода списков
31.	Выпадающий список. Список заполняется через метод, передающий строку	Поле для ввода даты (формат даты должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий объект
32.	Список с выбором. Список заполняется через метод, передающий строку	Поле для заполнения текста с проверкой, что длина введенного текста находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий объект
33.	Список значений. Список заполняется через метод, передающий строку	Поле для ввода вещественного значения с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий объект
34.	Выпадающий список. Список заполняется через публичное свойство	Поле для ввода номера телефона (номер телефона должен соответствовать шаблону)	Список значений. Список заполняется через метод, передающий объект, номер строки и имя свойства/поля
35.	Список с выбором. Список заполняется через публичное свойство	Поле для заполнения числа с проверкой, что число находится в определенном диапазоне	Дерево значений. Дерево заполняется через метод, передающий объект и имя свойства/поля
36.	Список значений. Список заполняется через публичное свойство	Поле для ввода даты (формат DD.MM.YYYY) с возможностью указания значения null	Таблица значений. Таблица заполняется через метод, передающий объект, номер строки и номер столбца

ЛАБОРАТОРНАЯ РАБОТА №2.
СОЗДАНИЕ НЕ ВИЗУАЛЬНОГО КОМПОНЕНТА

Цель

Научится создавать не визуальные компоненты и встраивать их в другие проекты.

Задание

1. *Создать описание компонентов, что должны получать на вход, что должно быть на выходе.*
2. *Создать библиотеку, в которой реализовать не визуальные компоненты по заданию.*
3. *Реализовать десктопное приложение (Windows Forms), использовать там разработанные компоненты.*

Решение

По заданию будет 3 не визульных компонента, которые должны использоваться на формах.

Пример. Не визуальный компонент отправки сообщений на почту

Рассмотрим задачу создания не визуального компонента для отправки сообщений на почту.

Компонент должен иметь публичное свойство для добавления адреса почты, с которой будут следовать отправки сообщений, и метод для рассылки сообщений. Отдельный метод нужен для добавления адресов рассылки. Создадим перечисления для результата отправки писем (листинг 2.1). Сделаем также свойство для возвзарщения текста ошибки, в случае неудачной отправки. Создадим библиотеку ClassLibraryControlMail для перечисления и компонента.

```
    public enum StatusResult
    {
        EmailsSend,

        NoSubject,

        NoMessage,
```

```
        NoAddressees,

        HaveError
    }
```

Листинг 2.1 – Перечисление StatusResult

Создадим компонент ComponentMailSender для отправки на почту (листинг 2.2).

```
    public partial class ComponentMailSender : Component
    {
        private string _mail;

        private string _password;

        private string _error;

        private List<string> emails;

        private Regex _reg = new Regex(pattern);

        private const string pattern = @"^(?("")(""[^""]+?""@)|(([0-9a-z]((\.(?!\.))|[-
!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=[0-9a-z])@))" +
              @"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]*[0-9a-z]*\.)+[a-z0-
9]{2,17}))$";

        public ComponentMailSender()
        {
            InitializeComponent();
        }

        public ComponentMailSender(IContainer container)
        {
            container.Add(this);

            InitializeComponent();
        }

        /// <summary>
        /// Адрес электронной почты, с которой будут отправляться письма
        /// </summary>
        public string Mail
```

```
        {
            get { return _mail; }
            set { if (!_reg.IsMatch(value)) { _mail = value; } }
        }

        /// <summary>
        /// Пароль электронной почты, с которой будут отправляться письма
        /// </summary>
        public string Password
        {
            get { return _password; }
            set { _password = value; }
        }

        /// <summary>
        /// Ошибка при отправке сообщений
        /// </summary>
        public string GetError { get { return _error; } }

        /// <summary>
        /// Добавление адреса в рассылку
        /// </summary>
        /// <param name="mail"></param>
        public void AddAddressee(string mail)
        {
            if (emails == null)
            {
                emails = new List<string>();
            }
            if (_reg.IsMatch(mail))
            {
                emails.Add(mail);
            }
        }

        public StatusResult SendMessage(string subject, string message)
        {
            MailMessage objMailMessage = new MailMessage();
            SmtpClient objSmtpClient = null;

            if (string.IsNullOrEmpty(subject))
            {
```

```
            return StatusResult.NoSubject;
        }
        if (string.IsNullOrEmpty(message))
        {
            return StatusResult.NoMessage;
        }
        if (emails == null || emails.Count == 0)
        {
            return StatusResult.NoAddressees;
        }

        try
        {
            objMailMessage.From = new MailAddress(_mail);
            for (int i = 0; i < emails.Count; ++i)
            {
                objMailMessage.To.Add(new MailAddress(emails[i]));
            }
            objMailMessage.Subject = subject;
            objMailMessage.Body = message;
            objMailMessage.SubjectEncoding = Encoding.UTF8;
            objMailMessage.BodyEncoding = Encoding.UTF8;

            objSmtpClient = new SmtpClient("smtp.gmail.com", 587);
            objSmtpClient.UseDefaultCredentials = false;
            objSmtpClient.EnableSsl = true;
            objSmtpClient.DeliveryMethod = SmtpDeliveryMethod.Network;
            objSmtpClient.Credentials = new NetworkCredential(_mail, _password);

            objSmtpClient.Send(objMailMessage);

            return StatusResult.EmailsSend;
        }
        catch (Exception ex)
        {
            _error = ex.Message;
            return StatusResult.HaveError;
        }
        finally
        {
            objMailMessage = null;
            objSmtpClient = null;
```

```
            }
        }
    }
```

Листинг 2.2 – Компонент ComponentMailSender

Подключим библиотеку к тестовму проекту с первой лабораторной, добавим на форму компонент (рисунок 2.1).

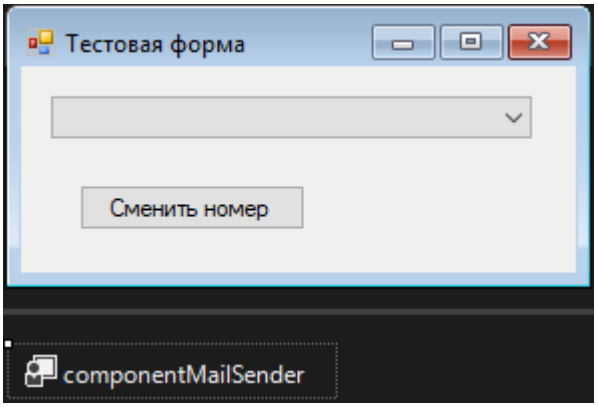

Рисунок 2.1 – Добавление компонента на форму

В свойствах компонента можно прописать логин и пароль (рисунок 2.2).

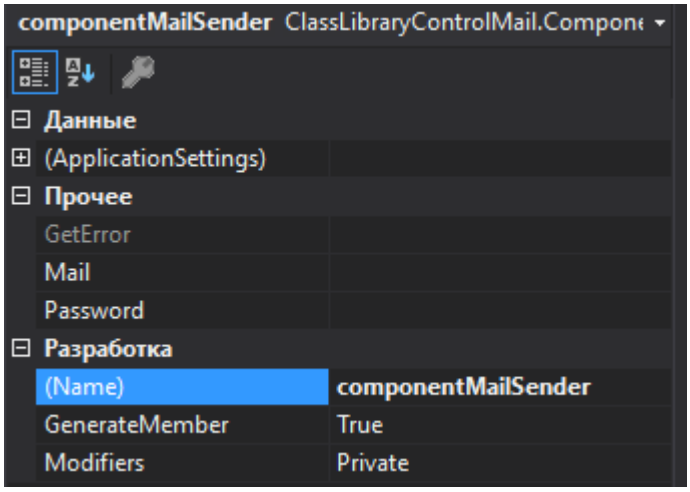

Рисунок 2.2 – Свойства компонента componentMailSender

Для теста в логике формы надо вызвать методы добавления адреса, добавить туда тестовый адерс и отправить на него сообщение.

Все компоненты разделены на 3 блока: выбора, ввода и вывода списка.

Блоки не визуальных компонент:

- Не визуальный компонент работы с данными:

    - Не визуальный компонент для создания бекапа
    - Не визуальный компонент для восстановления бекапа
    - Не визуальный компонент для хранения данных
- Не визуальный компонент создания отчета
    - Не визуальный компонент создания отчета в Word
    - Не визуальный компонент создания отчета в Excel
    - Не визуальный компонент создания отчета в PDF
- Не визуальный компонент создания диаграмм
    - Не визуальный компонент создания диаграмм в Word
    - Не визуальный компонент создания диаграмм в Excel
    - Не визуальный компонент создания диаграмм в PDF

Каждый из группы выбирает по одному компоненту из блоков. Получается, каждому нужно будет реализовать 3 компонента: один компонент выбора, один компонент ввода и один компонент вывода списка.

Не визуальные компоненты

1. Сохранение данных (backup). Данные сохраняются в файл в формате json. Проверять, что тип передаваемых данных настроен для сохранения в формате json (у класса указан требуемый атрибут). Данные для сохранения передаются через публичный метод. Файл архивируется.
2. Сохранение данных (backup). Данные сохраняются в бинарный файл. Проверять, что тип передаваемых данных настроен для сохранения в бинарном формате (у класса указан требуемый атрибут). Данные для сохранения передаются через публичный метод. Файл архивируется.
3. Сохранение данных (backup). Данные сохраняются в файл в формате xml. Проверять, что тип передаваемых данных настроен для сохранения в формате xml (у класса указан требуемый атрибут).

Данные для сохранения передаются через публичный метод. Файл архивируется.

4. Загрузка данных (restore). Данные загружаются из файла формата json. Проверять, что тип передаваемых данных настроен для для загрузки из формата json (у класса указан требуемый атрибут). Данные возвращаются через публичный метод. Файл лежит в архиве.
5. Загрузка данных (restore). Данные загружаются из бинарного файла. Проверять, что тип передаваемых данных настроен для загрузки из бинарного формата (у класса указан требуемый атрибут). Данные возвращаются через публичный метод. Файл лежит в архиве.
6. Загрузка данных (restore). Данные загружаются из файла формата xml. Проверять, что тип передаваемых данных настроен для загрузки из формата xml (у класса указан требуемый атрибут). Данные возвращаются через публичный метод. Файл лежит в архиве.
7. Хранилище данных (store). Данные хранятся в файле формата json. Проверять, что тип передаваемых данных настроен для работы с форматом json (у класса указан требуемый атрибут). Есть два публичных метода – получить данные и сохранить данные.
8. Хранилище данных (store). Данные хранятся в бинарном файле. Проверять, что тип передаваемых данных настроен для работы с бинарным форматом (у класса указан требуемый атрибут). Есть два публичных метода – получить данные и сохранить данные.
9. Хранилище данных (store). Данные хранятся в файле формата xml. Проверять, что тип передаваемых данных настроен для работы с форматом xml (у класса указан требуемый атрибут). Есть два публичных метода – получить данные и сохранить данные.
10. Отчет Word. Таблица с объединением колонок в шапке. Шапка записывается в первой строке. Данные передаются через публичный метод. Отдельно указываются номера колонок для объединения в шапке таблицы.

11. Отчет Word. Таблица с объединением строк в шапке. Шапка записывается в первой колонке. Данные передаются через публичный метод. Отдельно указываются номера строк для объединения в шапке таблицы.
12. Отчет Word. Таблица, у которой шапка либо первая строка, либо первая колонка. Данные передаются через публичный метод. Отдельно указывается шапка таблицы (первая колонка или первая строка).
13. Отчет Excel. Таблица с объединением колонок в шапке. Шапка записывается в первой строке. Данные передаются через публичный метод. Отдельно указываются номера колонок для объединения в шапке таблицы.
14. Отчет Excel. Таблица с объединением строк в шапке. Шапка записывается в первой колонке. Данные передаются через публичный метод. Отдельно указываются номера строк для объединения в шапке таблицы.
15. Отчет Excel. Таблица, у которой шапка либо первая строка, либо первая колонка. Данные передаются через публичный метод. Отдельно указывается шапка таблицы (первая колонка или первая строка).
16. Отчет PDF. Таблица с объединением колонок в шапке. Шапка записывается в первой строке. Данные передаются через публичный метод. Отдельно указываются номера колонок для объединения в шапке таблицы.
17. Отчет PDF. Таблица с объединением строк в шапке. Шапка записывается в первой колонке. Данные передаются через публичный метод. Отдельно указываются номера строк для объединения в шапке таблицы.
18. Отчет PDF. Таблица, у которой шапка либо первая строка, либо первая колонка. Данные передаются через публичный метод.

Отдельно указывается шапка таблицы (первая колонка или первая строка).

19. Диаграмма в Word. Тип диаграммы – линейная. Данные передаются через публичный метод. Отдельно задаются подписи к графику.
20. Диаграмма в Word. Тип диаграммы – гистограмма. Данные передаются через публичный метод. Отдельно задаются подписи к графику.
21. Диаграмма в Word. Тип диаграммы – круговая. Данные передаются через публичный метод. Отдельно задаются подписи к графику.
22. Диаграмма в Excel. Тип диаграммы – линейная. Данные передаются через публичный метод. Отдельно задаются подписи к графику.
23. Диаграмма в Excel. Тип диаграммы – гистограмма. Данные передаются через публичный метод. Отдельно задаются подписи к графику.
24. Диаграмма в Excel. Тип диаграммы – круговая. Данные передаются через публичный метод. Отдельно задаются подписи к графику.
25. Диаграмма в PDF. Тип диаграммы – линейная. Данные передаются через публичный метод. Отдельно задаются подписи к графику.
26. Диаграмма в PDF. Тип диаграммы – гистограмма. Данные передаются через публичный метод. Отдельно задаются подписи к графику.
27. Диаграмма в PDF. Тип диаграммы – круговая. Данные передаются через публичный метод. Отдельно задаются подписи к графику.

Варианты

№ вар.	Номер копмонента работы с данными	Номер компонента создания отчета	Номер компонента создания диаграммы
1.	1	13	25
2.	4	16	19
3.	7	10	22
4.	2	14	26
5.	5	17	20
6.	8	11	23
7.	3	15	27
8.	6	18	21
9.	9	12	24
10.	1	13	25
11.	4	16	19
12.	7	10	22
13.	2	14	26
14.	5	17	20
15.	8	11	23
16.	3	15	27
17.	6	18	21
18.	9	12	24
19.	1	13	25
20.	4	16	19
21.	7	10	22
22.	2	14	26
23.	5	17	20
24.	8	11	23
25.	3	15	27
26.	6	18	21
27.	9	12	24
28.	1	13	25

№ вар.	Номер копмонента работы с данными	Номер компонента создания отчета	Номер компонента создания диаграммы
29.	4	16	19
30.	7	10	22
31.	2	14	26
32.	5	17	20
33.	8	11	23
34.	3	15	27
35.	6	18	21
36.	9	12	24

ЛАБОРАТОРНАЯ РАБОТА №3.
РАЗРАБОТКА ПРОГРАММЫ С ИСПОЛЬЗОВАНИЕМ КОМПОНЕНТ

Цель

Научится применять созданные компоненты в процессе разработки проектов.

Задание

1. *Выбрать компоненты, которе понадобятся для реализации проекта (не использовать свои компоненты, разработанные в рамках предыдущих проектов, не более 2-х компонентов от одного автора).*
2. *Дать оценку используемым компонентам: удобство использования, стабильность работы, информативность методов.*
3. *Реализовать десктопное приложение (Windows Forms), согласно заданию.*

Общие требования

Приложение должно состоять из 2-3 модулей (отдельно логика, отдельно визуальное представление). Данные должны храниться в БД (MS SQL Server).

При разработке визуального представления использовать следующие наработки:

- компоненты для хранения данных / создания бекапа / восстановления данных из архива;
- компоненты для создания отчетов и графиков;
- компоненты для вывода списка на экран;
- компоненты для ввода данных;
- компонента для выбора;

Задания по вариантам

1. Учет успеваемости студентов. По студенту хранить следующую информацию: ФИО, курс (выбор из выпадающего списка), стипендия (может не получать). Выводить в виде дерева (по курсам). Создавать

бекап по данным. Формировать отчет в Word по студентам с курсами. Диаграмма в Word, сколько студентов какую стипендию получают.

2. Учет успеваемости студентов. По студенту хранить следующую информацию: ФИО, форма обучения (выбор из общего списка), электронная почта. Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по студентам с их почтами. Диаграмма в Excel, сколько студентов по какой форме обучаются.
3. Учет успеваемости студентов. По студенту хранить следующую информацию: ФИО, выбранные направления при поступлении (выбор из списка значений), дата поступления (диапазон). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по студентам с выбранными направлениями. Диаграмма в PDF, сколько студентов в какие года поступили.
4. Учет книг в библиотеке. По книге хранить следующую информацию: название, жанр (выбор из выпадающего списка), стоимость (может быть бесплатной). Выводить в виде дерева (по жанрам). Создавать бекап по данным. Формировать отчет в Word по книгам с жанрами. Диаграмма в Word, сколько книг какой стоимости.
5. Учет книг в библиотеке. По книге хранить следующую информацию: название, форма книги (выбор из общего списка), дата (формат dd month yyyy). Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по книгам с датами. Диаграмма в Excel, сколько книг в какой форме изданы.
6. Учет книг в библиотеке. По книге хранить следующую информацию: название, авторы (выбор из списка значений), аннотация (обязательное заполнение). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по книгам с аннотациями. Диаграмма в PDF, сколько книг какие авторы написали.
7. Учет сотрудников предприятия. По сотруднику хранить следующую информацию: ФИО, должность (выбор из выпадающего списка), дата

отпуска в текущем году (может еще не был). Выводить в виде дерева (по должностям). Создавать бекап по данным. Формировать отчет в Word по сотрудникам с должностями. Диаграмма в Word, сколько сотрудников в каждом месяце отдыхали.

8. Учет сотрудников предприятия. По сотруднику хранить следующую информацию: ФИО, подразделение (выбор из общего списка), рабочий телефон. Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по сотрудникам с их телефонами. Диаграмма в Excel, сколько сотрудников в каждом отделе.
9. Учет сотрудников предприятия. По сотруднику хранить следующую информацию: ФИО, навыки, которыми владеет (выбор из списка значений), стаж работы в фирме (диапазон). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по сотрудникам с их навыками. Диаграмма в PDF, сколько сотрудников какой стаж имеют.
10. Учет клиентов магазина. По клиенту хранить следующую информацию: ФИО, статус клиента (выбор из выпадающего списка), сумму покупок (может не быть). Выводить в виде дерева (по статусам). Создавать бекап по данным. Формировать отчет в Word по клиентам со статусами. Диаграмма в Word, сколько клиентов на какую сумму совершили покупок.
11. Учет клиентов магазина. По клиенту хранить следующую информацию: ФИО, пункт выдачи (выбор из общего списка), электронная почта. Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по клиентам с их почтами. Диаграмма в Excel, сколько клиентов в каких пунктах забирают посылки.
12. Учет клиентов магазина. По клиенту хранить следующую информацию: ФИО, выбранные категории товаров (выбор из списка значений), дата регистрации (диапазон). Выводить в виде списка.

Использовать хранилище данных. Формировать отчет в PDF по клиентам с выбранными категориями. Диаграмма в PDF, сколько клиентов в какие года регистрировались.

13. Учет продуктов в магазине. По продуктам хранить следующую информацию: название, категория изделия (выбор из выпадающего списка), количество на складке (может не быть). Выводить в виде дерева (по категориям). Создавать бекап по данным. Формировать отчет в Word по продуктам с категориями. Диаграмма в Word, сколько продуктов в каком количестве есть на складе.
14. Учет продуктов в магазине. По продуктам хранить следующую информацию: название, единицы измерения поставок (упак, кг, литры и т.п.) (выбор из общего списка), дата поставки (формат dd month yyyy). Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по продуктам с датами поставки. Диаграмма в Excel, сколько продуктов в каких единицах поставлюятся.
15. Учет продуктов в магазине. По продуктам хранить следующую информацию: название, производители (выбор из списка значений), данные по СанПиН (обязательное заполнение). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по продуктам с данными по СанПиН. Диаграмма в PDF, сколько продуктов какие производители поставляют.
16. Учет поставщиков фабрики мебели. По поставщику хранить следующую информацию: название, тип организации (ИП, ООО и т.п.) (выбор из выпадающего списка), дата последней поставки в этом году (могло не быть). Выводить в виде дерева (по типам организации). Создавать бекап по данным. Формировать отчет в Word по поставщикам с их типами. Диаграмма в Word, сколько поставщиков в каждом месяце поставляли.

17. Учет поставщиков фабрики мебели. По поставщику хранить следующую информацию: название, ФИО менджера, который с ним работает (выбор из общего списка), рабочий телефон поставщика. Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по поставщикам с их телефонами. Диаграмма в Excel, сколько поставщиков у каждого менеджера.
18. Учет поставщиков фабрики мебели. По поставщику хранить следующую информацию: название, типы поставляемых изделий (выбор из списка значений), частота поставок в году (диапазон). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по поставщикам с поставлемыми ими типами изделий. Диаграмма в PDF, сколько поставщиков с какой частотой поставляют изделия.
19. Учет заказов интернет-магазина. По заказу хранить следующую информацию: номер, статус заказа (выбор из выпадающего списка), сумму заказа (может не быть). Выводить в виде дерева (по статусам). Создавать бекап по данным. Формировать отчет в Word по заказам со статусами. Диаграмма в Word, сколько заказов на какую сумму было создано.
20. Учет заказов интернет-магазина. По заказу хранить следующую информацию: номер, город назначения (выбор из общего списка), электронная почта заказчика. Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по заказам с их почтами. Диаграмма в Excel, сколько заказов в какие города отправляются.
21. Учет заказов интернет-магазина. По заказу хранить следующую информацию: номер, выбранные товаров (выбор из списка значений), дата получения заказа (диапазон). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по

заказам с выбранными товарами. Диаграмма в PDF, сколько заказов в какие дни получались.

22. Учет счетов в кафе. По счету хранить следующую информацию: ФИО официанта, тип заказа (с собой, закузка, основное блюдо и т.п.) (выбор из выпадающего списка), сумма заказа (может не быть). Выводить в виде дерева (по типам заказа). Создавать бекап по данным. Формировать отчет в Word по счетам с категориями. Диаграмма в Word, сколько счетов с какими суммами создавались.
23. Учет счетов в кафе. По счету хранить следующую информацию: ФИО официанта, номер столика (выбор из общего списка), дата заказа (формат dd month yyyy). Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по счетам с датами заказа. Диаграмма в Excel, сколько счетов на какие столики приходятся.
24. Учет счетов в кафе. По счету хранить следующую информацию: ФИО официанта, блюда (выбор из списка значений), данные по клиенту (обязательное заполнение). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по счетам с данными по клиенту. Диаграмма в PDF, в скольких счетах какие блюда фигурируют.
25. Учет доставок курьерской службы. По доставке хранить следующую информацию: ФИО курьера, тип доставки (экспресс, 1 класса и т.п.) (выбор из выпадающего списка), дата доставки (могло не быть). Выводить в виде дерева (по типам доставок). Создавать бекап по данным. Формировать отчет в Word по доставкам с их типами. Диаграмма в Word, сколько доставок в каждом месяце поставляли.
26. Учет доставок курьерской службы. По доставке хранить следующую информацию: ФИО курьера, офис доставки (выбор из общего списка), рабочий телефон офиса. Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по доставкам с их телефонами. Диаграмма в Excel, сколько доставок у каждого офиса.

27. Учет доставок курьерской службы. По доставке хранить следующую информацию: ФИО курьера, что доставляется (уведомления, письма, посылки и т.п.) (выбор из списка значений), количество поставляемой продукции (диапазон). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по доставкам с поставками. Диаграмма в PDF, сколько доставок с каким количеством производятся.
28. Учет аккаунтов портала. По аккаунту хранить следующую информацию: логин, роль (выбор из выпадающего списка), количество посещений (может не быть). Выводить в виде дерева (по ролям). Создавать бекап по данным. Формировать отчет в Word по аккаунтам с ролями. Диаграмма в Word, сколько аккаунтов сколько раз посещали портал.
29. Учет аккаунтов портала. По аккаунту хранить следующую информацию: логин, город проживания (выбор из общего списка), электронная почта пользователя. Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по аккаунтам с их почтами. Диаграмма в Excel, сколько аккаунтов в каких городах проживают.
30. Учет аккаунтов портала. По аккаунту хранить следующую информацию: логин, выбранные интересы (выбор из списка значений), дата создания аккаунта (диапазон). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по аккаунтам с выбранными интересами. Диаграмма в PDF, сколько аккаунтов в какие дни созданы.
31. Учет лабораторных работ. По лабораторной хранить следующую информацию: тема лабораторной, сложность работы (выбор из выпадающего списка), средний балл сдававших (может не быть). Выводить в виде дерева (по сложности). Создавать бекап по данным. Формировать отчет в Word по лабораторным с их сложностями.

Диаграмма в Word, сколько лабораторных с какими средними баллами сдавались.

32. Учет лабораторных работ. По лабораторной хранить следующую информацию: тема лабораторной, дисциплина (выбор из общего списка), дата выдачи (формат dd month yyyy). Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по лабораторным с датами выдачи. Диаграмма в Excel, сколько лабораторных по каким дисциплинам сдаются.
33. Учет лабораторных работ. По лабораторной хранить следующую информацию: тема лабораторной, ФИО сдающих (выбор из списка значений), результат сдачи (обязательное заполнение). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по лабораторным с данными по сдачи. Диаграмма в PDF, в скольких лабораторных какие студенты фигурируют.
34. Учет подразделений организации. По подразделению хранить следующую информацию: ФИО руководителя, тип подразделения (выбор из выпадающего списка), дата отчета в этом году (могло не быть). Выводить в виде дерева (по типам подразделений). Создавать бекап по данным. Формировать отчет в Word по подразделениям с их типами. Диаграмма в Word, сколько подразделений в каждом месяце отчитывались.
35. Учет подразделений организации. По подразделению хранить следующую информацию: ФИО руководителя, головное подразделение (выбор из общего списка), рабочий телефон подразделения. Выводить в виде таблицы. Загрузка данных из бекапа. Формировать отчет в Excel по подразделениям с их телефонами. Диаграмма в Excel, сколько подразделений у каждого головного подразделения.
36. Учет подразделений организации. По подразделению хранить следующую информацию: ФИО руководителя, профессии

сотрудников (выбор из списка значений), количество сотрудников в подразделении (диапазон). Выводить в виде списка. Использовать хранилище данных. Формировать отчет в PDF по подразделениям с сотрудниками. Диаграмма в PDF, сколько подразделений с каким количеством сотрудников.

ЛАБОРАТОРНАЯ РАБОТА №4.
ПРИМЕНЕНИЕ ПОРОЖДАЮЩИХ ПАТТЕРНОВ

Цель

Научиться применять порождающие паттерны в процессе проектирования системы.

Задание

1. *Придумать формулировки задач на основе задачи предыдущей лабораторной, в рамках решения которых потребовалось бы применить паттерн (на каждый паттерн –* задачу*).*
2. *Создать отдельный компонент, реализующий указанный паттерн.*
3. *Применить компонент в разработанной системе.*

Решение

Рассмотрим задачу учета успеваемости студентов. В рамках задачи требуеся фиксировать то, как студент посещает занятия и как выполняет необходимую на занятии работу. Допустим уже есть система, которая позводяет создать студента и внести свдения по студенту, в том числе средний балл по успеваемости. Придумаем как изменить эту задачу, чтобы можно было применить паттерн Abstract Factory.

Паттерн Abstract Facotry

Допустим, для учета успеваемости требуется учитывать посещаемость занятий и оценки, получаемые студеном на занятии. Занятия могут быть нескольких типов, например, лекция, практика, лабораторная, семинар и т.п. Здесь и можно применить паттерн, для создания занятия.

Основной класс будет отвечать за занятие. У него есть свойство с названием занятия, свойство с описанием и свойство-объект асбтрактного класса для расчета оценки за занятие студентом. Этот класс состоит из 2 объектов классов: класс-посещение, реализцющий функционал учета, как идет фиксация посещаемости занятия и класс-сдача, отвечающий за фиксацию выполнения заданий занятия. У классов посещаемости и фиксации выполнения должны быть методы, возвращающие числовые значения (балл

за посещение и за выполнение). Они могут, например, принимать булевское значение, был ли студент на занятии.

От абстрактного класса сделаем 3 реализации:

- класс с учетом посещения и сдачи;
- класс с учетом только посещаемости;
- класс с учетом только сдачи заданий.

Также потребуется абстрактный класс, отвечающий за создание занятия и несколько его реализаций под разные типы занятий (ограничимся тремя: лекция, практика, лабораторная).

В основном классе сделаем конструктор, который бы принимал название, описание, а также объект абстрактного класса-фабрики и метод, для получения оценки, исходя из того, был ли студент на занятии и выполнил ли он задание (могут передаваться как параметры в метод).

Сделаем визуальный компонент для создания объекта-занятия. Компонент должен уметь возвращать запись (отдельный класс) с описанием занятия (название и описание) и балл студента за занятие.

Архитектура решения представлена на рисунке 4.1.

Все это реализуется в виде отдельной библиотеки и подключается в существующую систему.

Теперь в системе при занесении информации об успеваемости студента мы будем вызывать форму с компонентом по заданию и через него получать балл за занятие.

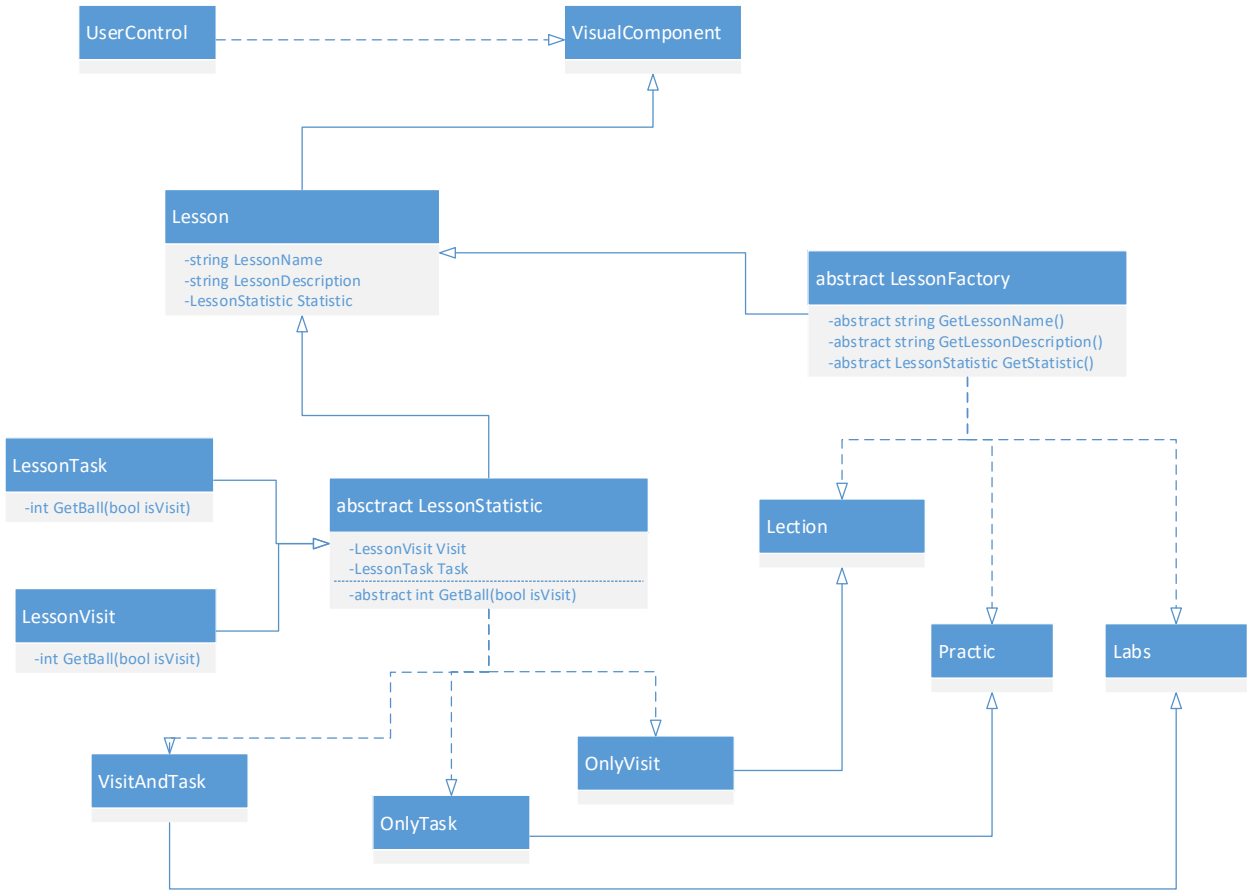

Рисунок 4.1 –Диаграмма классов для паттерна Abstract Factory

Паттерны по вариантам

1. AbstractFactory, Builder. Реализовать паттерн AbstractFactory.
2. AbstractFactory, FactoryMethod. Реализовать паттерн AbstractFactory.
3. AbstractFactory, Prototype. Реализовать паттерн AbstractFactory.
4. AbstractFactory, Singleton. Реализовать паттерн AbstractFactory.
5. AbstractFactory, ObjectPool. Реализовать паттерн AbstractFactory.
6. Builder, FactoryMethod. Реализовать паттерн Builder.
7. Builder, Prototype. Реализовать паттерн Builder.
8. Builder, Singleton. Реализовать паттерн Builder.
9. Builder, ObjectPool. Реализовать паттерн Builder.
10. FactoryMethod, Prototype. Реализовать паттерн FactoryMethod.
11. FactoryMethod, Singleton. Реализовать паттерн FactoryMethod.
12. FactoryMethod, ObjectPool. Реализовать паттерн FactoryMethod.
13. Prototype, Singleton. Реализовать паттерн Prototype.

14. Prototype, ObjectPool. Реализовать паттерн Prototype.
15. Singleton, ObjectPool. Реализовать паттерн Singleton.
16. AbstractFactory, Builder. Реализовать паттерн Builder.
17. AbstractFactory, FactoryMethod. Реализовать паттерн FactoryMethod.
18. AbstractFactory, Prototype. Реализовать паттерн Prototype.
19. AbstractFactory, Singleton. Реализовать паттерн Singleton.
20. AbstractFactory, ObjectPool. Реализовать паттерн ObjectPool.
21. Builder, FactoryMethod. Реализовать паттерн FactoryMethod.
22. Builder, Prototype. Реализовать паттерн Prototype.
23. Builder, Singleton. Реализовать паттерн Singleton.
24. Builder, ObjectPool. Реализовать паттерн ObjectPool.
25. FactoryMethod, Prototype. Реализовать паттерн Prototype.
26. FactoryMethod, Singleton. Реализовать паттерн Singleton.
27. FactoryMethod, ObjectPool. Реализовать паттерн ObjectPool.
28. Prototype, Singleton. Реализовать паттерн Singleton.
29. Prototype, ObjectPool. Реализовать паттерн ObjectPool.
30. Singleton, ObjectPool. Реализовать паттерн ObjectPool.

ЛАБОРАТОРНАЯ РАБОТА №5.
ПРИМЕНЕНИЕ СТРУКТУРНЫХ ПАТТЕРНОВ

Цель

Научиться применять структурные паттерны в процессе проектирования системы.

Задание

1. *Придумать формулировки задач на основе задачи предыдущей лабораторной, в рамках решения которых потребовалось бы применить паттерн (на каждый паттерн –* задачу*).*
2. *Создать отдельный компонент, реализующий указанный паттерн.*
3. *Применить компонент в разработанной системе.*

Решение

Рассмотрим задачу учета успеваемости студентов. В рамках задачи требуеся фиксировать то, как студент посещает занятия и как выполняет необходимую на занятии работу. Допустим уже есть система, которая позводяет создать студента и внести свдения по студенту, в том числе средний балл по успеваемости. Придумаем как изменить эту задачу, чтобы можно было применить паттерн Adapter.

Паттерн Adapter

Допустим существует система работы с сотрудниками, позволяющая нанимать новых сотрудников, фиксировать их работу (выполнение задач, поставленных перед ними) и уволнять их. У нас есть интерефейс, который содержит методы для добавления студента, учета его успеваемости и удаления студента. Создадим адаптер между интерфейсами. Логика будет следующей: вместо добавления (зачисления) студента будет прием сотрудника, вместо результата посещения занятий – результат проделанной работы, вместо удаления (отчисление) студента – увольнение.

Для реализации нам потребуется создать интерфейс по работе с сотрудником и адаптер для связи с интерфейсом работы со студентами.

Также нужен будет класс – сотрудник. Интерфейс по работе со студентом уже должен быть в рамках 3 лабораторной.

Создадим визуальный компонент для работы с сотрудниками, при этом сохраненине будет происходить в существующую базу, в таблицу студентов.

Архитектура решения будет следующей (рисунок 5.1):

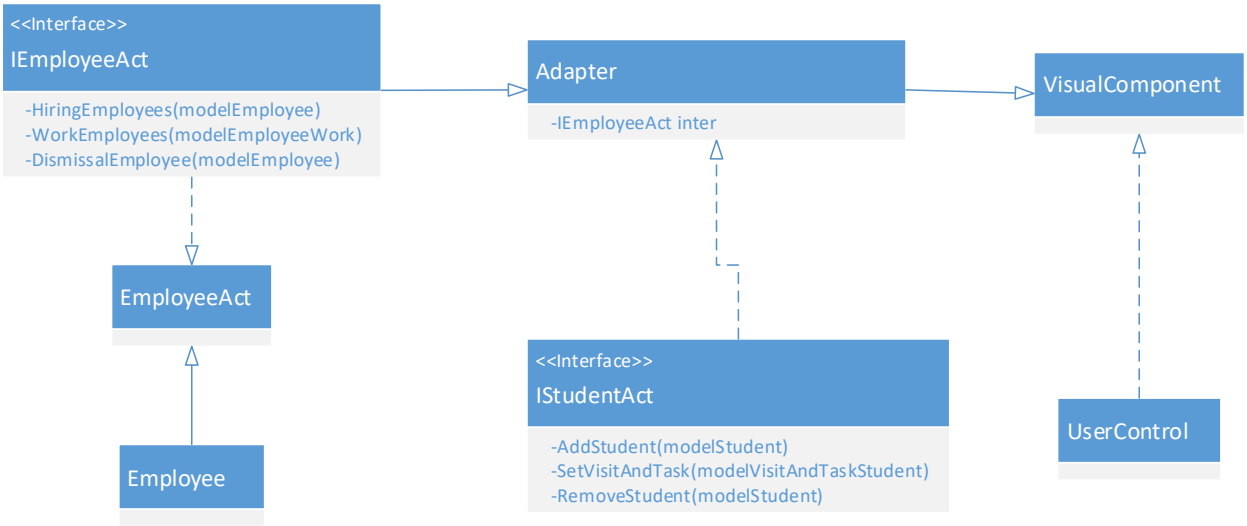

Рисунок 5.1 – Диаграмма классов для паттерна Adapter

Все это реализуется в виде отдельной библиотеки и подключается в существующую систему.

Теперь в системе можно работать как со студентами, так и с сотрудниками.

Паттерны по вариантам

1. Adapter, Bridge. Реализовать паттерн Adapter.
2. Adapter, Composite. Реализовать паттерн Adapter.
3. Adapter, Decorator. Реализовать паттерн Adapter.
4. Adapter, Facade. Реализовать паттерн Adapter.
5. Adapter, Proxy. Реализовать паттерн Adapter.
6. Bridge, Composite. Реализовать паттерн Bridge.
7. Bridge, Decorator. Реализовать паттерн Bridge.
8. Bridge, Facade. Реализовать паттерн Bridge.
9. Bridge, Proxy. Реализовать паттерн Bridge.
10. Composite, Decorator. Реализовать паттерн Composite.

11. Composite, Facade. Реализовать паттерн Composite.
12. Composite, Proxy. Реализовать паттерн Composite.
13. Decorator, Facade. Реализовать паттерн Decorator.
14. Decorator, Proxy. Реализовать паттерн Decorator.
15. Facade, Proxy. Реализовать паттерн Facade.
16. Adapter, Bridge. Реализовать паттерн Bridge.
17. Adapter, Composite. Реализовать паттерн Composite.
18. Adapter, Decorator. Реализовать паттерн Decorator.
19. Adapter, Facade. Реализовать паттерн Facade.
20. Adapter, Proxy. Реализовать паттерн Proxy.
21. Bridge, Composite. Реализовать паттерн Composite.
22. Bridge, Decorator. Реализовать паттерн Decorator.
23. Bridge, Facade. Реализовать паттерн Facade.
24. Bridge, Proxy. Реализовать паттерн Proxy.
25. Composite, Decorator. Реализовать паттерн Decorator.
26. Composite, Facade. Реализовать паттерн Facade.
27. Composite, Proxy. Реализовать паттерн Proxy.
28. Decorator, Facade. Реализовать паттерн Facade.
29. Decorator, Proxy. Реализовать паттерн Proxy.
30. Facade, Proxy. Реализовать паттерн Proxy.

ЛАБОРАТОРНАЯ РАБОТА №6.
ПРИМЕНЕНИЕ ПАТТЕРНОВ ПОРОЖДЕНИЯ

Цель

Научиться применять паттерны порождения в процессе проектирования системы.

Задание

1. *Придумать формулировки задач на основе задачи предыдущей лабораторной, в рамках решения которых потребовалось бы применить паттерн (на каждый паттерн –* задачу*).*
2. *Создать отдельный компонент, реализующий указанный паттерн.*
3. *Применить компонент в разработанной системе.*

Решение

Рассмотрим задачу учета успеваемости студентов. В рамках задачи требуеся фиксировать то, как студент посещает занятия и как выполняет необходимую на занятии работу. Допустим уже есть система, которая позводяет создать студента и внести свдения по студенту, в том числе средний балл по успеваемости. Придумаем как изменить эту задачу, чтобы можно было применить паттерн ChainOfResponsibility.

Паттерн ChainOfResponsibility

Допустим требуется система для учета приема экзамена у студентов. При этом экзаменаторов может быть несколько. Каждый приходящий на экзамен проходит по преподавателям, если есть свободный, то он начинает сдавать ему экзамен (преподаватель станвоится недоступным на определенный прмоежуток времени).

Для реализации нам потребуется создать абстрактный класс преподавателя, у которого будет свойство-объект от этого же класса (ссылка на следующего экзаменатора), свойство – затраичваемое время на студента, поле – типа поток, абстрактный метод – принять студента. 3 класса-наследника от абстрактного класса: профессор, доцент, ассистент. У каждого

будет свое время, затрачиваемое на студента, для профессора минимальное, для ассистента – максимальное.

Методы принятия по логике будет следующим: проверяется, свободен ли поток или нет, если он свободен, то он запускается (для этого нужен доп метод, в котором вызывается только команда Sleep на то время, которе требуется преподавателю на прием студента), если нет, то передает прием студента следующему за ним экзаменатору, если он есть. Метод должен возвращать строку – результат сдачи экзамена.

Создадим визуальный контрол для приема экзамена. В нем будет задаваться список экзаменаторов (с произвольным количеством), приниматься студент и отправляться одному из экзаменаторов. Результат будет выводится на экран (ListBox).

Архитектура решения будет следующей:

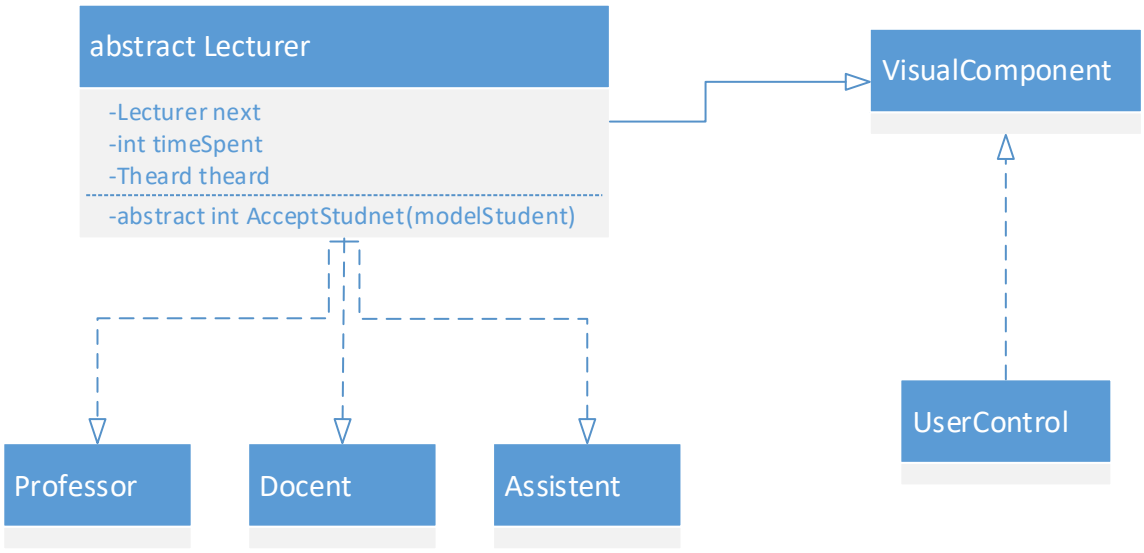

Рисунок 6.1 – Диграмма классов для паттерна ChainOfResponsibility

Все это реализуется в виде отдельной библиотеки и подключается в существующую систему.

Теперь в системе можно принимать экзамены у студентов.

Паттерны по вариантам

1. ChainOfResponsibility, Command, Interpreter, Mediator. Реализовать паттерн ChainOfResponsibility.

2. Command, Interpreter, Mediator, Memento. Реализовать паттерн Command.
3. Interpreter, Mediator, Memento, Observer. Реализовать паттерн Interpreter.
4. Mediator, Memento, Observer, State. Реализовать паттерн Mediator.
5. Memento, Observer, State, Strategy. Реализовать паттерн Memento.
6. Observer, State, Strategy, TemplateMethod. Реализовать паттерн Observer.
7. State, Strategy, TemplateMethod, Visitor. Реализовать паттерн State.
8. Strategy, TemplateMethod, Visitor, ChainOfResponsibility. Реализовать паттерн Strategy.
9. TemplateMethod, Visitor, ChainOfResponsibility, Command. Реализовать паттерн TemplateMethod.
10. Visitor, ChainOfResponsibility, Command, Interpreter. Реализовать паттерн Visitor.
11. ChainOfResponsibility, Memento, Observer, State. Реализовать паттерн ChainOfResponsibility.
12. Command, Observer, State, Strategy. Реализовать паттерн Command.
13. Interpreter, State, Strategy, TemplateMethod. Реализовать паттерн Interpreter.
14. Mediator, Strategy, TemplateMethod, Visitor. Реализовать паттерн Mediator.
15. Memento, TemplateMethod, Visitor, ChainOfResponsibility. Реализовать паттерн Memento.
16. Observer, Visitor, ChainOfResponsibility, Command. Реализовать паттерн Observer.
17. State, ChainOfResponsibility, Command, Interpreter. Реализовать паттерн State.
18. Strategy, Command, Interpreter, Mediator. Реализовать паттерн Strategy.

19. TemplateMethod, Interpreter, Mediator, Memento. Реализовать паттерн TemplateMethod.
20. Visitor, Mediator, Memento, Observer. Реализовать паттерн Visitor.
21. ChainOfResponsibility, Strategy, TemplateMethod, Visitor. Реализовать паттерн ChainOfResponsibility.
22. Command, TemplateMethod, Visitor, ChainOfResponsibility. Реализовать паттерн Command.
23. Interpreter, Visitor, ChainOfResponsibility, Command. Реализовать паттерн Interpreter.
24. Mediator, ChainOfResponsibility, Command, Interpreter. Реализовать паттерн Mediator.
25. Memento, Command, Interpreter, Mediator. Реализовать паттерн Memento.
26. Observer, Interpreter, Mediator, Memento. Реализовать паттерн Observer.
27. State, Mediator, Memento, Observer. Реализовать паттерн State.
28. Strategy, Memento, Observer, State. Реализовать паттерн Strategy.
29. TemplateMethod, Observer, State, Strategy. Реализовать паттерн TemplateMethod.
30. Visitor, State, Strategy, TemplateMethod. Реализовать паттерн Visitor.

ЛАБОРАТОРНАЯ РАБОТА №7.
РАЗРАБОТКА ПЛАГИНОВ

Цель

Научится создавать плагины и применять их в проектах.

Задание

1. *Разработать интерфейс под логику, описанную в задании.*
2. *Создать плагин для одной из задач, получить плагины (1-2) от одногруппников.*
3. *Внедрить плагины в проект.*

Решение

Рассмотрим задачу учета успеваемости студентов. В рамках задачи требуеся фиксировать то, как студент посещает занятия и как выполняет необходимую на занятии работу.

Применение плагинов

Дополним функционал программы за счет плагинов. Сделаем ряд действий над студентом, помимо учета посещаемости и сдачи работ на занятии:

1. Выполнение курсовой работы.
2. Прохождение практики.
3. Сдача экзамена.

Создадим проект-библиотеку для описания конфигурации плагинов. Первым делом создадим классы-модели для данных, передаваемых в плагины. У нас будет один такой класс с информацией по студенту. Плагин простой, так что достаточно передавать Id студента (считаем, что все плагины и главная система работают с единой БД). Определим общий интерфейс для плагина. Он будет состоять из следующих свойств и методов:

- свойство получения название плагина (для вывода на форму для выбора), возвращает строку;
- метод для выполнения функционала над студентом (принимает объект от класса-связи, ничего не возвращает).

Следующий шаг – разработка самих плагинов. Создаем проект-библиотеку, добавляем туда библиотеку (файл с расширением *.dll) с описанием конфигурации плагина. В проекте создаем реализацию от интерфейса плагина. Готовую библиотеку (файл с расширением *.dll) возвращаем заказчку (или в свой основной проект).

В основном проекте создаем класс-менеджер, который будет искать и подключать плагины, а также форму на которой будет выводится список студнтов, выпадающий список (или ListBox, или иной вариант вывода) с названиями найденных плагинов и кнопка вызова метода плагина. При желании, интерфейс для плагина можно изменить, сделав у метода выполнения функционала возвращаемое значение с результатом выполнения операции.

Замечание. Третий пункт плагинов подразумевает форимрование документа с возможностью его печати.

Плагины по вариантам

1. Учет успеваемости студентов. Плагины:
    a. перевод на другой курс;
    b. назначение стипендии;
    c. формирвоание справки по студенту.
2. Учет успеваемости студентов. Плагины:
    a. перевод на иную форму обучения;
    b. отправка уведомления на почту;
    c. формирование приказа на смену фамилии студента.
3. Учет успеваемости студентов. Плагины:
    a. смена выбранных направлений обучения;
    b. перезачисление студента;
    c. формирование приказа на зачисление.
4. Учет книг в библиотеке. Плагины:
    a. смена жанра книги;
    b. изменение стоимости книги;

c. формирование ценника книги.

5. Учет книг в библиотеке. Плагины:
    a. изменение формата книги;
    b. переиздание книги;
    c. формирование артикула книги;
6. Учет книг в библиотеке. Плагины:
    a. изменение списка авторов;
    b. редактирование аннотации;
    c. формирование вкладки описания книги.
7. Учет сотрудников предприятия. Плагины:
    a. перевод сотрудника на новую должность;
    b. отправка сотрудника в отпуск;
    c. формирование приказа на отпуск.
8. Учет сотрудников предприятия. Плагины:
    a. перевод сотрудника в новое подразделение;
    b. смена номера телефона сотрудника;
    c. формирование приказа на перевод сотрудника в новое подразделение.
9. Учет сотрудников предприятия. Плагины:
    a. прохождение обучения сотрудником;
    b. увеличение стажа сотрудника;
    c. формирование приказа о надбавке за стаж.
10. Учет клиентов магазина. Плагины:
    a. смена статуса клиента на основе суммы покупок;
    b. изменение суммы покупок;
    c. формирование чека.
11. Учет клиентов магазина. Плагины:
    a. изменение пункта выдачи товара;
    b. оповещение клиента о поступлении товара;
    c. формирвоание накладной на выдачу.

12. Учет клиентов магазина. Плагины:
    a. добавление интересующих категорий;
    b. перерегистрация клиента;
    c. формирование карточки клиента.
13. Учет продуктов в магазине. Плагины:
    a. изменение категории изделия;
    b. пополнение продукта на складке;
    c. формрование накладной на выдачу продукта.
14. Учет продуктов в магазине. Плагины:
    a. изменение единицы измерения поставок;
    b. поступление продукта;
    c. формирование накладной на поставку (без учета количества).
15. Учет продуктов в магазине. Плагины:
    a. добавление новго производителя;
    b. изменения требований по СанПиН;
    c. формирования справки по продукту.
16. Учет поставщиков фабрики мебели. Плагины:
    a. смена типа организации;
    b. поступление товара;
    c. формирование накладной на поставку.
17. Учет поставщиков фабрики мебели. Плагины:
    a. назначение нового менеджера поставщику;
    b. изменение рабочего телефона поставщика;
    c. формирование карточки поставщика.
18. Учет поставщиков фабрики мебели. Плагины:
    a. добавление новых изделий поставщика;
    b. изменение частоты поставок;
    c. формирование поручения на поставку.
19. Учет заказов интернет-магазина. Плагины:
    a. перевод статуса заказа в новое состояние;

b. изменение суммы заказа;
c. формирование карточки заказа.

20. Учет заказов интернет-магазина. Плагины:
    a. смена города назначения;
    b. отправка уведомления заказчку о доставке;
    c. форимрование документа о заказе.
21. Учет заказов интернет-магазина. Плагины:
    a. изменение списка товаров;
    b. перенос даты получения товаров на n дней;
    c. формирование чека заказа.
22. Учет счетов в кафе. Плагины:
    a. изменение типа заказа;
    b. перерасчет суммы заказа;
    c. формирование листа заказа.
23. Учет счетов в кафе. Плагины:
    a. перевод посетителей за другой столик;
    b. изменение даты заказа;
    c. формирование карты заказа.
24. Учет счетов в кафе. Плагины:
    a. добавление блюда в заказ;
    b. изменение сведений по клиенту;
    c. формирование заказа на кухню.
25. Учет доставок курьерской службы. Плагины:
    a. изменение типа доставки;
    b. перенос даты поставки на n дней;
    c. формирование наклейки на поставку.
26. Учет доставок курьерской службы. Плагины:
    a. смена офиса доставки;
    b. изменение рабочего телеофна офиса;
    c. формирование бланка доставки курьеру.

27. Учет доставок курьерской службы. Плагины:
    a. добавление в доставку элементов;
    b. изменение количества доставляемой продукции;
    c. формирование накладной клиенту.
28. Учет аккаунтов портала. Плагины:
    a. изменение роли пользователя;
    b. фиксация посещения;
    c. формирование справки по аккаунту.
29. Учет аккаунтов портала. Плагины:
    a. перезд в другой город;
    b. отправка уведомления пользователю;
    c. формирование справки по перезду.
30. Учет аккаунтов портала. Плагины:
    a. добавление новых интересов;
    b. перерегистрация;
    c. формирования списка предпочтений пользователя.
31. Учет лабораторных работ. Плагины:
    a. изменение сложности работы;
    b. расчет среднего балла;
    c. формирование отчета по лабораторной.
32. Учет лабораторных работ. Плагины:
    a. дублирование лабораторной в другую дисциплину;
    b. перевыкладование лабораторной;
    c. формирование справки по выдачи лабораторной.
33. Учет лабораторных работ. Плагины:
    a. отметка о сдаче студентом лабораторной;
    b. изменени результата сдачи;
    c. формирование отчета по сдаче лабораторной.
34. Учет подразделений организации. Плагины:
    a. изменение типа подразделения;

b. прием отчета от подразделения;
c. формирование отчета.

35. Учет подразделений организации. Плагины:
    a. переподчинение подразделения;
    b. изменение рабочего телефона;
    c. формирование приказа на переодчинение подразделения.

36. Учет подразделений организации. Плагины:
    a. расширение сферы знаний подразделения;
    b. прием новых сотруднков;
    c. формирование карточки подразделения.

ЛАБОРАТОРНАЯ РАБОТА №8.
СОЗДАНИЕ УСТАНОВОЧНОГО ФАЙЛА

Цель

Научиться создавать установщик для разворачивания своего приложения на других устройствах.

Задание

1. *Создать установщик.*
2. *Проверить установку на компьютере.*
3. .

Список использованных источников

1. Pro Git. 2nd Edition [Электронный ресурс] / Режим доступа: https://git-scm.com/book/ru/v2.
2. METANIT.COM. Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com/sharp/. – Загл. с экрана.
3. ProfessorWeb. .Net & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru/. – Загл. с экрана.
4. Tiberiu Covaci, Rod Stephens, Vincent Varallo, Gerry O'Brien. MCSD Certification Toolkit (Exam 70-483) // Published by John Wiley & Sons, Inc. – 2013. – 656p.
5. MCTS Self-Paced Training Kit (Exam 70-536): Microsoft .NET Framework–Application Development Foundation, Second Edition eBook // Published by Microsoft Press. – 2009. – 829 p.