МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

Высшего профессионального образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

ОБНИНСКИЙ ИНСТИТУТ АТОМНОЙ ЭНЕРГЕТИКИ

Отделение ИКС

**Проектное задание**

по дисциплине “Облачные сервисы и технологии”

Паттерны:

«Шаблонный метод. Посетитель»

Выполнила:

студентка гр. ИВТ-М20 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Илюшкина Е.Н.

Проверил:

доцент кафедры КССТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тельнов В.П.

Обнинск, 2021г.

Содержание

1. Постановка задачи2

[1.1 Цель работы4](#_Toc69578234)

[1.2 Формулировка задачи4](#_Toc69578235)

[2. Паттерны проектирования5](#_Toc69578236)

[2.1 Шаблонный метод5](#_Toc69578237)

[2.1.1 Основные характеристики5](#_Toc69578238)

[2.1.2. Решение6](#_Toc69578239)6

[2.2 Посетитель6](#_Toc69578240)

[2.2.1 Основные характеристики6](#_Toc69578241)

[2.2.2. Решение8](#_Toc69578242)8

[3. Спецификация требований9](#_Toc69578243)9

[3.1 Диаграмма совокупных требований9](#_Toc69578244)9

[3.2 Диаграмма системных требований10](#_Toc69578245)10

[3.3 Диаграмма требований к интерфейсу11](#_Toc69578246)11

[3.4 Диаграмма функциональных требований12](#_Toc69578247)2

[3.5 Прочие диаграммы требований13](#_Toc69578248)3

[3.6 Диаграмма прецедентов14](#_Toc69578249)4

[3.7 Сценарии выполнения прецедентов15](#_Toc69578250)5

[4. Технический проект17](#_Toc69578251)6

[4.1 Диаграмма классов18](#_Toc69578252)6

[4.1.1 Шаблонный метод18](#_Toc69578253)6

[4.1.2 Посетитель18](#_Toc69578254)6

[4.2 Диаграммы последовательности19](#_Toc69578255)

[4.2.1 Получение информации19](#_Toc69578256)

[4.2.2 Получение статистики19](#_Toc69578257)19

[4.2.3 Получение суммы покупок20](#_Toc69578258)0

[4.3 Диаграммы компонентов20](#_Toc69578259)0

[4.4 Диаграммы коммуникации21](#_Toc69578260)1

[4.4.1 Получение информации21](#_Toc69578261)1

[4.4.2 Получение статистики22](#_Toc69578262)2

[4.2.3 Получение суммы покупок23](#_Toc69578263)3

[4.5 Диаграмма развёртывания24](#_Toc69578264)4

[5. Пользовательский интерфейс25](#_Toc69578265)5

[5.1 Шаблонный метод 25](#_Toc69578266)5

[5.2 Посетитель25](#_Toc69578269)5

[6. Тестирование26](#_Toc69578275)26

[6.1 Сценарии тестирования, тестовые условия и наборы данных26](#_Toc69578276)26

[6.2 Протокол тестирования27](#_Toc69578277)27

[6.3 Результаты тестирования и рефакторингаОшибка! Закладка не определена.](#_Toc69578278)28

[6.3.1 Используемые метрики процесса тестирования28](#_Toc69578279)28

[6.3.2 Рефакторинг кода28](#_Toc69578280)28

[28](#_Toc69578281)29

[29](#_Toc69578282)29

[29](#_Toc69578283)29

[29](#_Toc69578284)29

[29](#_Toc69578285)29

[29](#_Toc69578286)29

[30](#_Toc69578287)30

[30](#_Toc69578288)30

[7.5 Решение известных проблем (Troubleshooting30](#_Toc69578289)30

[8. Листинг 32](#_Toc69578290)32

[8.1 app.py32](#_Toc69578291)32

[8.2 template\_method.py34](#_Toc69578292)34

[8.3 visitor.py35](#_Toc69578293)35

[8.4 base.html38](#_Toc69578294)38

[8.5 index.html40](#_Toc69578295)40

8.6 template\_method.html4040

[8.7 visitor.html41](#_Toc69578297)

# 1. Постановка задачи

## 1.1 Цель работы

Требуется для двух паттернов проектирования софта (Шаблонный метод и Посетитель) реализовать на языке Python работоспособное веб-приложение, размещенное на облачном хостинге. Пример должен быть сделан на языке программирования Python с использованием фреймворка Flask.

## 1.2 Формулировка задачи

Проектируемое приложение: «Система просмотра статистики поступающих в учебные заведения и затраты на основные товары к началу учебного года».

Поддерживаемые данные:

* Общие данные
  + Название учебных заведений;
  + Путь запуска скрипта для анализа всех данных.
* Статистика
  + Поступившие ученики/студенты;
  + Отсеянные ученики/студенты.
* Сумма покупок
  + Тетради;
  + Канцтовары.

Поддерживаемые операции:

* Получение статистики за период подачи документов в учебные заведения;
* Получение суммы основных покупок к началу учебного года.

# 2. Паттерны проектирования

## 2.1 Шаблонный метод

Шаблонный метод (Template Method) – это поведенческий паттерн проектирования, который определяет скелет алгоритма, перекладывая ответственность за некоторые его шаги на подклассы. Паттерн позволяет подклассам переопределять шаги алгоритма, не меняя его общей структуры.

Паттерн особенно важен для создания библиотек классов, поскольку предоставляют возможность вынести общее поведение в библиотечные классы.

### 2.1.1 Основные характеристики

**Паттерн используется, когда:**

* Планируется, что в будущем подклассы должны будут переопределять различные этапы алгоритма без изменения его структуры;
* В классах реализующим схожий алгоритм, происходит дублирование кода. Вынесение общего кода в шаблонный метод уменьшит его дублирование на подклассах;
* Подклассы должны расширять базовый алгоритм, не меняя его структуры.

**Участники паттерна:**

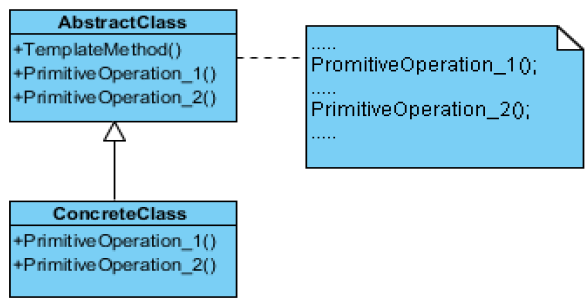
Класс AbstractClass – абстрактный класс:

* определяет абстрактные примитивные операции, замещаемые в конкретных классах-потомках для реализации шагов алгоритма;
* реализует шаблонный метод, определяющий основу (каркас, скелет) алгоритма. Шаблонный метод вызывает примитивные операции, а также операции, определенные в классе AbstractClass или в других объектах.

Класс ConcreteClass – конкретный класс:

* реализует примитивные операции, выполняющие шаги алгоритма способом, который зависит от данного конкретного класса-потомка.

### 2.1.2. Решение



## 

## 2.2 Посетитель

Посетитель (Visitor) — Это поведенческий паттерн проектирования, который позволяет добавлять в программу новые операции, не изменяя классы объектов, над которыми эти операции могут выполняться.

### 2.2.1 Основные характеристики

**Паттерн используется, когда:**

* Нужно выполнить какую-то операцию над всеми элементами сложной структуры объектов. Посетитель позволяет применять одну и ту же операцию к объектам различных классов;
* Когда над объектами сложной структуры объектов надо выполнять некоторые не связанные между собой операции, но вы не хотите «засорять» классы такими операциями. Посетитель позволяет извлечь родственные операции из классов, составляющих структуру объектов, поместив их в один класс-посетитель. Если структура объектов является общей для некоторых приложений, то паттерн позволяет в каждое приложение включить только нужные операции;
* Когда новое поведение имеет смысл только для некоторых классов из существующей иерархии. Посетитель позволяет определить поведение только для этих классов, оставив его пустым для всех остальных.

**Участники паттерна:**

Класс Visitor – посетитель:

* объявляет операцию Visit для каждого класса ConcreteElement в структуре объектов. Сигнатура этой операции идентифицируют класс, который посылает посетителю запрос Visit. Это позволяет посетителю «узнать», элемент какого конкретного класса он посещает. Владея такой информацией, посетитель может обращаться к элементу напрямую через его интерфейс.

Класс ConcreteVisitor – конкретный посетитель:

* реализует все операции, объявленные в классе Visitor. Каждая операция реализует фрагмент алгоритма, определенного для класса соответствующего объекта в структуре. Класс ConcreteVisitor предоставляет контекст для этого алгоритма и сохраняет его локальное состояние. Часто в этом состоянии аккумулируются результаты, полученные в процессе обхода структуры.

Класс Element – элемент:

* объявляет операцию Visit для каждого класса ConcreteElement в структуре объектов. Сигнатура этой операции идентифицируют класс, который посылает посетителю запрос Visit. Это позволяет посетителю «узнать», элемент какого конкретного класса он посещает. Владея такой информацией, посетитель может обращаться к элементу напрямую через его интерфейс.

Класс ConcreteElement – конкретный элемент:

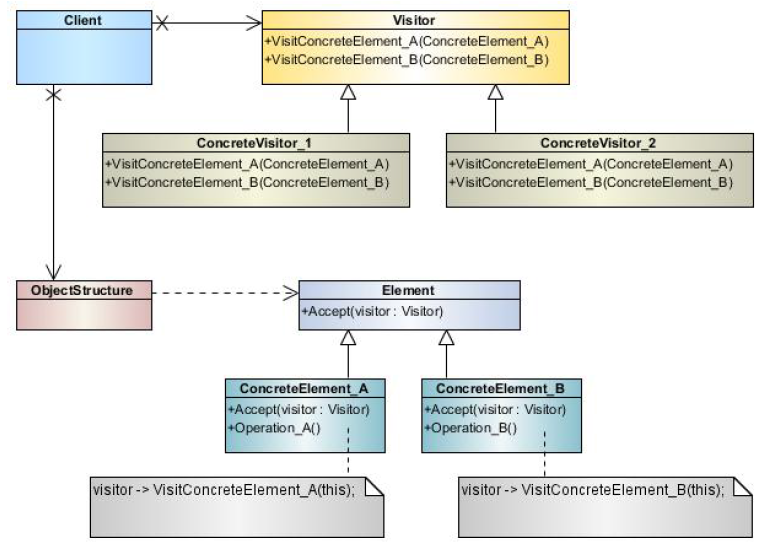
* реализует операцию Accept, принимающую посетителя как аргумент.

Класс ObjectStructure – структура объектов:

* может перечислить свои элементы.
* может предоставить посетителю высокоуровневый интерфейс для посещения своих элементов.
* может быть составным объектом (см. паттерн «Компоновщик») или коллекцией.

### 

### 2.2.2. Решение



**3. Спецификация требований**

## 3.1 Диаграмма совокупных требований

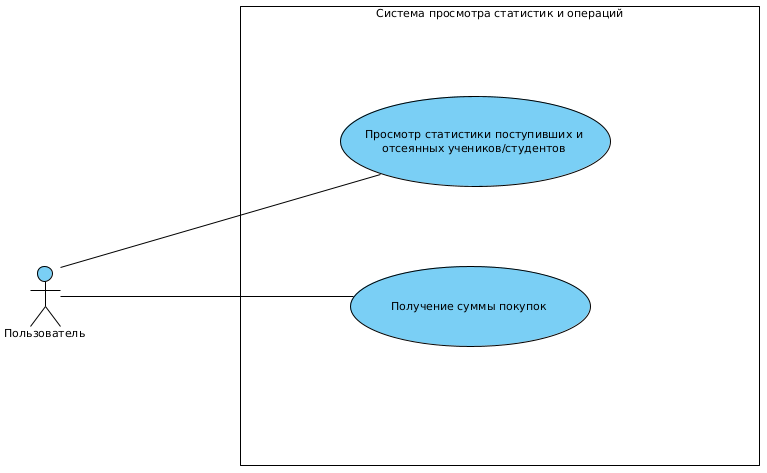
## C:\Users\user\Desktop\ghjn.jpg3.2 Диаграмма системных требований

## 3.3 Диаграмма требований к интерфейсу

## 3.4 Диаграмма функциональных требований

## 3.5 Прочие диаграммы требований

## 3.6 Диаграмма прецедентов



Пользователь, зайдя в систему может получить информацию о количестве зачисленных и отсеянных учеников/студентов.

Дополнительно пользователю доступна возможность оценить сумму затрат на основные товары для учебы.

## 3.7 Сценарии выполнения прецедентов

№1 Раздел «Общее описание сценария»

|  |  |
| --- | --- |
| Прецедент | Получение статистики поступающих в учебные заведения |
| Акторы | Пользователь |
| Цель | Просмотр пользователем мониторинга информации о кол-ве зачисленных и отсеянных учеников |
| Краткое описание | Пользователь отправляет на сервер соответствующий запрос |
| Тип | Базовый прецедент |
| Ссылки на другие прецеденты | - |

Раздел «Типичный ход событий»

|  |  |
| --- | --- |
| Действия акторов | Отклик системы |
| 1. Пользователь, находясь в системе отправляет запрос на получение информации | 2. Система отображает название учебного заведения и статистику по поступающим. |

№2 Раздел «Общее описание сценария»

|  |  |
| --- | --- |
| Прецедент | Оценка суммы на покупке для обучения в зависимости он учебного заведения |
| Акторы | Пользователь |
| Цель | Получить необходимую сумму за покупки |
| Краткое описание | Пользователь отправляет на сервер соответствующий запрос |
| Тип | Базовый прецедент |
| Ссылки на другие прецеденты | - |

Раздел «Типичный ход событий»

|  |  |
| --- | --- |
| Действия акторов | Отклик системы |
| 1. Пользователь, находясь в системе отправляет запрос на получение информации | 2. Система отображает сумму основных покупок в зависимости от учебного заведения |

# 4. Технический проект

Классы: School (Школа), University (Институт), TestVisitorSchool (Сумма покупок для школы), TestVisitorUniversity (Сумма покупок для института) . Опишем каждый класс отдельно.

* Класс «Школа» содержит:
  1. **Методы**

1. **демонстрация() –** вернуть текущие значение поступивших в учебное заведение;
2. **демонстрация() –** вернуть текущие значение отсеянных в учебное заведение.

* Класс «Институт» содержит:
  1. **Методы**

1. **демонстрация() –** вернуть текущие значение поступивших в учебное заведение;
2. **демонстрация() –** вернуть текущие значение отсеянных в учебное заведение.

* Класс «Сумма покупок для школы» содержит:

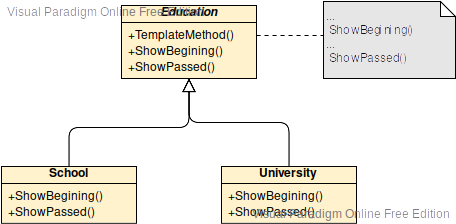
1. **Лист покупок** –– поле private типа “ String”;
2. **Сумма покупок** –– поле private типа “ String”;
3. **Методы**
4. **операция() –** вернуть текущие значение суммы;
5. **конструктор() –** конструирует список покупок и сумму;
6. **перебор() –** проходит по каждому элементу списка и передает в калькулятор.

* Класс «Сумма покупок для института» содержит:

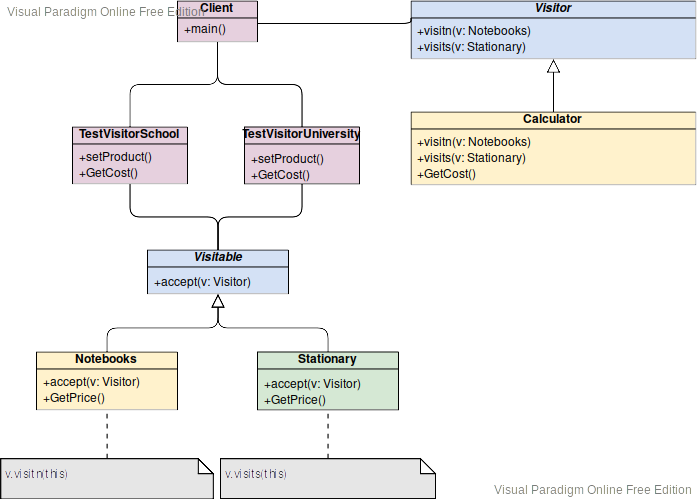
1. **Лист покупок** –– поле private типа “ String”.
2. **Сумма покупок** –– поле private типа “String”.
3. **Методы**
4. **операция()** – вернуть текущие значение суммы;
5. **конструктор() –** конструирует список покупок и сумму;
6. **перебор() –** проходит по каждому элементу списка и передает в калькулятор.

## 4.1 Диаграмма классов

### 4.1.1 Шаблонный метод



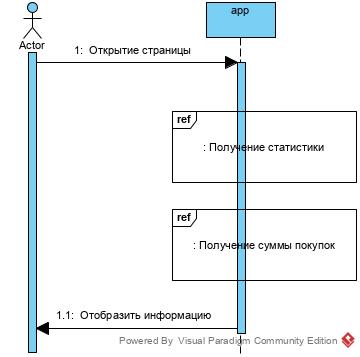
### 4.1.2 Посетитель



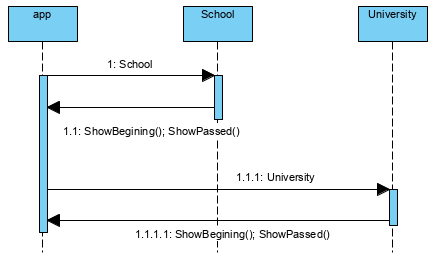
## 

## 4.2 Диаграммы последовательности

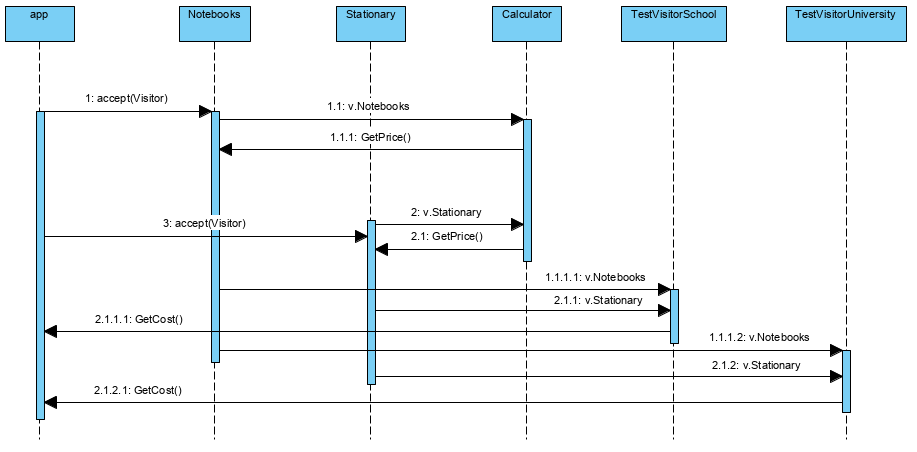
### 4.2.1 Получение информации



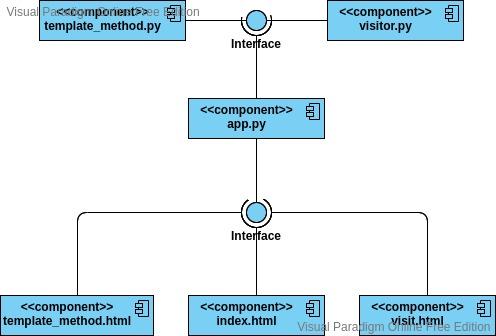
### 4.2.2 Получение статистики



### 4.2.3 Получение суммы покупок

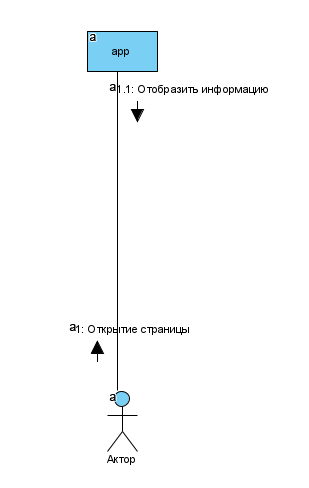


## 4.3 Диаграммы компонентов

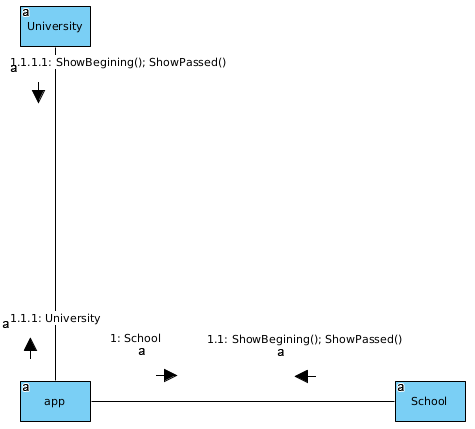
****

## 4.4 Диаграммы коммуникации

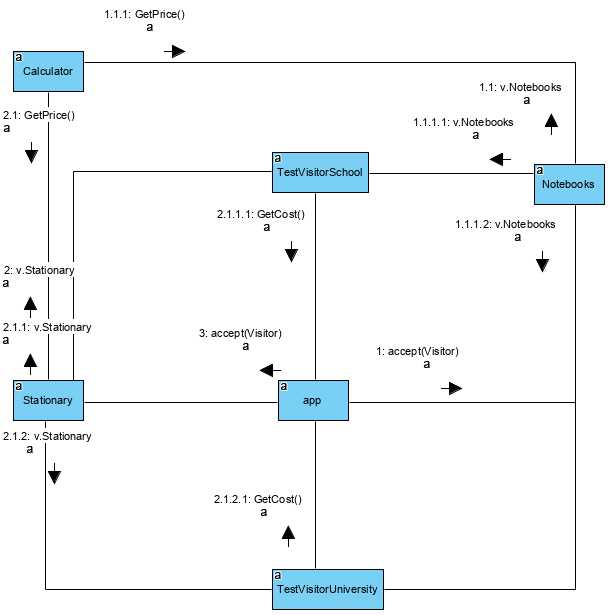
### 4.4.1 Получение информации



### 4.4.2 Получение статистики

****

### 4.4.3 Получение суммы покупок



## C:\Users\user\Desktop\Тельнов\hfpdthnsdfybt.png4.5 Диаграмма развёртывания

# 5. Пользовательский интерфейс

## 5.1 Шаблонный метод

**5.2 Посетитель**

## 

# 6. Тестирование

Выполнить комплексное автоматизированное тестирование программного решения.

Выполнить расчет метрик тестирования. Метрики процесса тестирования по стандарту IEEE 982.1-1998

Продумать варианты улучшения внешнего вида кода, компоновки папок проекта, оптимизации программного решения (рефакторинг).

## 6.1 Сценарии тестирования, тестовые условия и наборы данных

I. На основании pytest-тестов проверить работоспособность написанных классов по отдельности

1. Класс School.py

* Проверка отображение статистики по поступившим и отсеянным в школу методами ShowBegining() и ShowPassed();

1. Класс University.py

* Проверка отображение статистики по поступившим и отсеянным в ВУЗ методами ShowBegining() и ShowPassed();

1. Класс TestVisitorSchool.py

* Проверка отображение суммы покупок для школы методами setProduct() и GetCost();

1. Класс TestVisitorUniversity.py

* Проверка отображение суммы покупок для ВУЗа методами setProduct() и GetCost();

II. Через веб-интерфейс протестировать работоспособность проекта в целом

1. Проверить на орфографические и пунктуационные ошибки.
2. Проверить визуальное отображение элементов.
3. Проверить правильность переходов по страницам приложения.

III. Протестировать работоспособность на разных компьютерах и браузерах

1. Проверить работу в различных браузерах

* Internet Explorer
* Google Chrome
* Mozilla Firefox
* Opera

2. Проверить быстродействие на разных компьютерах

* Процессор 8 ядер 3.2 GHz, 8 Gb оперативной памяти
* Процессор 2 ядра 1.80 GHz, 4 Gb оперативной памяти

## 6.2 Протокол тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Класс** | **Наименование тестирования** | **Ожидаемый результат** | **Результаты теста** |
| 1 | School.py | Отображение количества поступивших и отсеянных | Количество отображено | Успешен |
| 2 | University.py | Отображение количества поступивших и отсеянных | Количество отображено | Успешен |
| 3 | TestVisitorSchool.py | Отображение потраченной суммы на тетради и канцтовары | Сумма отображена | Успешен |
| 4 | TestVisitorUniversity.py | Отображение потраченной суммы на тетради и канцтовары | Сумма отображена | Успешен |

## 6.3 Результаты тестирования и рефакторинга

### 6.3.1 Используемые метрики процесса тестирования

1. **IEEE 1**. Плотность отказов = [Число уникальных отказов, найденных при тестировании] / [Число строк кода]:

2. **IEEE 18**. Надёжность работы выражается вероятностью того, что в k произвольных случаях работы программа вернёт корректный результат. Эта величина оценивается через выполнение некоторого числа запусков программы (N) и вычисление числа случаев успешной работы (S). Вероятность успеха, таким образом, вычисляется как S/N, а вероятность возможности отработать k раз успешно – как произведение вероятностей каждого успешного запуска, то есть [S/N]\*[ S/N]\*…\*[ S/N], или [S/N]^k. Входные данные для каждого запуска выбираются произвольно и независимо от предыдущего запуска.

3. **IEEE 22**. Оценка числа оставшихся отказов (методом засева). Эта оценка получена путем «засеивания» в программу N произвольных отказов. Если s - число найденных засеянных отказов, а f - число других отказов, найденных за тот же период тестирования, оценка равна f × N / s.

4. **IEEE 36**. Точность тестирования = f/N, где N – это число засеянных отказов, а f – это число засеянных ошибок, найденных во время тестирования.

### 6.3.2 Рефакторинг кода

Функционал определения данных о количестве поступивших/отсеянных и сумме покупок были вынесены в отдельный модуль. Большая часть способов получения данных метрик была переписана с использованием встроенных средств языка. В случае перехода на несуществующую страницу с использованием метода Error() было добавлено логирование адреса перехода.

# 7. Руководство пользователя

## 7.1 Область применения

Программное обеспечение «Система просмотра статистики и суммы» может использовать как:

* Возможность просмотра статистики кол-ва поступивших/отсеянных;
* Возможность просмотра результата выполнения суммирования;

## 7.2 Краткое описание возможностей

1. Возможность просмотра статистики кол-ва поступивших/отсеянных. Пользователю предоставлена возможность ознакомиться со значениями числа поступивших/отсеянных в учебные заведения. Для этого ему необходимо на любой странице, перейти по ссылке, располагающейся вверху экрана.
2. Возможность просмотра результата выполнения суммирования. Пользователю предоставлена возможность ознакомиться с результатами выполнения суммирования, выполненных в приложении. Для этого ему необходимо проделать те же шаги, что и при просмотре статистики.

## 7.3 Уровень подготовки пользователя

Конечный пользователь программы должен обладать практическими навыками работы в различных браузерах: IE8 и старше (в т.ч. Microsoft Edge), Google Chrome, Mozilla Firefox, Opera, Yandex browser. Обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы. Иметь представление о том, что такое Интернет-банкинг, примерно понимать его структуру.

## 7.4 Назначение программы

### 7.4.1 Виды деятельности, функции

1. Представление количества поступивших/отсеянных человек при поступлении в школу или ВУЗ.
2. Представление результата выполнения суммирования покупок для каждого учебного заведения отдельно.

### 7.4.2 Программные и аппаратные требования к системе

1. Поддерживаемые операционные системы: Windows XP, Windows 7, Windows 8/8.1, Windows 10;

2. Поддерживаемые браузеры: Google Chrome, IE8 и старше (в т.ч. Microsoft Edge), Opera, Mozilla Firefox, Yandex browser.

### 7.4.3 Подготовка к работе

Для начала работы с «Системой получения статистики поступающих в учебные заведения и затраты на основные товары к началу учебного года» необходимо открыть браузер (Mozilla Firefox, Google Chrome, IE8 и т.п), для обеспечения лучшей работоспособности рекомендуется использовать браузеры, описанные в п. 7.2 данного руководства.

В адресной строке необходимо ввести имя страницы, на которой представлена «Системой получения статистики поступающих в учебные заведения и затраты на основные товары к началу учебного года» в зависимости от сервера, на котором будет развернута программа.

Рекомендуемый адрес: http://

При вводе корректного адреса отобразится главная страница.

## 7.5 Решение известных проблем (Troubleshooting)

В ходе работы с «Система информации о состоянии сервера» могут возникнуть следующие ошибки:

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Ошибка** | **Описание ошибки** | **Требуемые действия пользователя при возникновении ошибки** |
| 1 | Сервер не найден | Возможны проблемы с сетью или с доступом к сайту приложения | Для устранения данной проблемы следует обратиться к администратору |

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Ошибка** | **Описание ошибки** | **Требуемые действия пользователя при возникновении ошибки** |
| 2 | Сайт отобразил 404 ошибка | Код ошибки соответствует спецификации «Not Found». Это означает, что страница не найдена, она перемещена или удалена | Для устранения данной проблемы следует проверить правильность набранного запроса в поисковой строке |
| 3 | Сайт отобразил 500 ошибка | Любая внутренняя ошибка сервера, которая не входит в рамки остальных ошибок класса | Для устранения данной проблемы следует связаться с администрацией сайта |
| 4 | Медленная загрузка сайта | Возможно, в данный момент ведутся технические работы на сайте или модифицируется контент | Для устранения данной проблемы следует воспользоваться сайт через некоторое время |

# 8. Листинг кода

## 8.1 app.py

from flask import Flask, render\_template  
  
from template\_method import (  
 School,  
 University  
)  
from visitor import (  
 TestVisitorSchool,  
 TestVisitorUniversity  
)  
  
app = Flask(\_\_name\_\_)  
  
  
def html(text: str) -> str:  
 return text.replace("\n", "<br/>").replace("\t", "&nbsp;&nbsp;&nbsp;&nbsp;")  
  
  
@app.route('/')  
def index():  
 return render\_template(  
 'index.html'  
 )  
  
  
@app.route('/template\_method')  
def template\_method():  
 school = School()  
 school.runAll()  
 print(school.runAll())  
  
 university = University()  
 university.runAll()  
 print(university.runAll())  
  
 return render\_template(  
 'template\_method.html',  
 template\_method1=html(school.runAll()),  
 template\_method2=html(university.runAll())  
 )  
  
  
@app.route('/visitor')  
def visitor():  
 s = TestVisitorSchool()  
 s.setProduct()  
 visitor1 = f"Всего к оплате за покупки к школе: {s.GetCost()}"  
 print(visitor1)  
  
 u = TestVisitorUniversity()  
 u.setProduct()  
 visitor2 = f"Всего к оплате за покупки к институту: {u.GetCost()}"  
 print(visitor2)  
  
 return render\_template(  
 'visitor.html',  
 visitor1=html(visitor1),  
 visitor2=html(visitor2)  
 )

if \_\_name\_\_ == "\_\_main\_\_":  
 app.run('127.0.0.1', debug=True)

## 8.2 template\_method.py

from abc import ABC, abstractmethod  
  
class Education(ABC):  
  
 # Абстрактный Класс определяет шаблонный метод, содержащий скелет некоторого алгоритма,  
 # состоящего из вызовов (обычно) абстрактных примитивных операций.  
 # Конкретные подклассы должны реализовать эти операции, но оставить сам шаблонный метод  
 # без изменений.  
  
 @abstractmethod  
 def ShowBegining(self) -> str:  
 pass  
  
 @abstractmethod  
 def ShowPassed(self) -> str:  
 pass  
  
 # Эти операции должны быть реализованы в подклассах.  
 def runAll(self):  
 print(self.ShowBegining)  
 print(self.ShowPassed)  
  
class School(Education):  
   
 # Конкретные классы должны реализовать все абстрактные операции базового  
 класса. Они также могут переопределить некоторые операции с реализацией по  
 умолчанию*.*  
  
# Эти операции уже имеют реализации.  
 @property  
 def ShowBegining(self) -> str:  
 return "в первый класс принято 90 человек"  
  
 @property  
 def ShowPassed(self) -> str:  
 return "в первом классе не приняли 12 человек"  
  
  
class University(Education):  
  
 @property  
 def ShowBegining(self) -> str:  
 return "в ВУЗ на бюджет зачислено 534 человека"  
  
 @property  
 def ShowPassed(self) -> str:  
 return "в ВУЗ на бюджет не прошли 175 человек"  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 school = School()  
 school.runAll()  
  
 university = University()  
 university.runAll()

## 8.3 visitor.py

from \_\_future\_\_ import annotations  
from abc import ABC, abstractmethod  
  
class Visitor(ABC):  
  
 @abstractmethod  
 def visitn(self, v:Notebooks) -> str:  
 pass  
  
 @abstractmethod  
 def visits(self, v:Stationary) -> str:  
 pass  
  
class Calculator(Visitor):  
  
 def \_\_init\_\_(self, cost:int = 0):  
 self.cost = cost  
  
 def visitn(self, v:Notebooks):  
 self.cost = self.cost + v.GetPrice()  
  
 def visits(self, v:Stationary):  
 self.cost = self.cost + v.GetPrice()  
  
 def GetCost(self) -> int:  
 return self.cost  
  
class Visitable(ABC):  
 #Интерфейс Компонента объявляет метод accept,  
 #который в качестве аргумента может получать любой объект,  
 #реализующий интерфейс посетителя.  
  
 @abstractmethod  
 def accept(self, v:Visitor):  
 pass  
  
class Notebooks(Visitable):  
 #Каждый Конкретный Компонент должен реализовать метод accept таким образом,  
 # чтобы он вызывал метод посетителя, соответствующий классу компонента.  
 def \_\_init\_\_(self, price: int):  
 self.price = price  
  
 def accept(self, v: Visitor):  
 v.visitn(self)  
  
 def GetPrice(self):  
 return self.price  
  
  
class Stationary(Visitable):  
  
 def \_\_init\_\_(self, price: int):  
 self.price = price  
  
 def accept(self, v: Visitor):  
 v.visits(self)  
  
 def GetPrice(self):  
 return self.price  
  
#Список покупок для школы:  
class TestVisitorSchool:  
  
 def \_\_init\_\_(self):  
 self.shopping\_list = [Notebooks(150), Stationary(567)]  
 self.calc = Calculator()  
  
 def setProduct(self):  
 for i in self.shopping\_list:  
 i.accept(self.calc)  
  
 def GetCost(self) -> int:  
 return self.calc.GetCost()  
  
#Список покупок для института:  
class TestVisitorUniversity:  
  
 def \_\_init\_\_(self):  
 self.shopping\_list = [Notebooks(175), Stationary(30)]  
 self.calc = Calculator()  
  
 def setProduct(self):  
 for i in self.shopping\_list:  
 i.accept(self.calc)  
  
 def GetCost(self) -> int:  
 return self.calc.GetCost()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 s = TestVisitorSchool()  
 s.setProduct()  
 print(f"Всего к оплате за покупки к школе: {s.GetCost()}")  
  
 u = TestVisitorUniversity()  
 u.setProduct()  
 print(f"Всего к оплате за покупки к институту: {u.GetCost()}")

## 8.4 base.html

<!doctype html>  
<html lang="ru">  
<head>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">  
 <title>{% block title %} {% endblock %}</title>  
</head>  
<body>  
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">  
 <div class="container-fluid">  
 <a class="navbar-brand" href="{{ url\_for('index') }}">Облачные сервисы и технологии</a>  
 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"  
 aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">  
 <span class="navbar-toggler-icon"></span>  
 </button>  
 <div class="collapse navbar-collapse" id="navbarNav">  
 <ul class="navbar-nav">  
 <li class="nav-item">  
 <a class="nav-link" href="{{ url\_for('template\_method') }}">Шаблонный метод</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link" href="{{ url\_for('visitor') }}">Посетитель</a>  
 </li>  
 </ul>  
 </div>  
 </div>  
</nav>  
<div class="container">  
 {% block content %} {% endblock %}  
</div>  
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js"  
 integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf"  
 crossorigin="anonymous"></script>  
</body>  
</html>

## 8.5 index.html

{% extends 'base.html' %}  
{% block title %} Облачные технологии и сервисы {% endblock %}  
{% block content %}  
 <div class="bg-light p-5 rounded">  
 <h1> Проектное задание</h1>  
 <br/>  
 <h2> Облачные сервисы и технологии</h2>  
 <br/>  
 <h3> Паттерны: Шаблонный метод. Посетитель</h3>  
 <br/>  
 <p>Группа: ИВТ-М20<br/>Cтудентка: Илюшкина Е.Н.</p>  
 </div>  
{% endblock %}

## 8.6 template\_method.html

{% extends 'base.html' %}  
{% block title %} Шаблонный метод {% endblock %}  
{% block content %}  
 <div class="bg-light p-5 rounded">  
 <h1>Шаблонный метод</h1>  
 <h2>Описание</h2>  
 <ul>В зависимости от учебного заведения получаем статистику поступивших и отсеянных человек. Учебные заведения:  
 <li>School</li>  
 <li>University</li>  
 </ul>  
 <h2>Результаты</h2>  
 <p class="lead">{{ template\_method1 | safe }}</p>

<p class="lead">{{ template\_method2 | safe }}</p>  
 </div>  
{% endblock %}

## 8.7 visitor.html

{% extends 'base.html' %}  
{% block title %} Посетитель {% endblock %}  
{% block content %}  
 <div class="bg-light p-5 rounded">  
 <h1>Посетитель</h1>  
 <h2>Описание</h2>  
 <ul>  
 Вычисление стоимости покупок (Notebooks, Stationary):  
 <li>Notebooks</li>  
 <li>Stationary</li>  
 </ul>  
 <h2>Сумма к оплате</h2>  
 <p class="lead">{{ visitor1 | safe }} </p>

<p class="lead">{{ visitor2 | safe }} </p>  
 </div>  
{% endblock %}