

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Спеціальність «Інформатика»

Звіт

Лабораторна робота № 3

З дисципліни «Об'єктно-орієнтоване програмування»

Варіант 6

Виконала: студентка 2 курсу

групи К-26

Абрамова Марія Вікторівна

“29” грудня 2016 р.

Київ – 2016

Зміст

| | |
|---|----|
| 1. Вступ..... | 3 |
| 2. Принцип роботи програми | 4 |
| 3. Структура XML-файлу | 9 |
| 4. Паттерн Стратегія | 11 |
| 4.1 LINQ..... | 13 |
| 4.2 DOM..... | 15 |
| 4.3 SAX | 18 |
| 4.4 Як вибрати між SAX і DOM | 20 |
| 5. Збереження результату аналізу даних..... | 21 |
| 6. XSL..... | 23 |
| 6.1 XPath | 23 |
| 6.2 XSLT | 24 |
| 7. HTML..... | 26 |
| 8. CSS..... | 28 |
| 9. Висновки | 33 |
| 10. Джерела | 34 |

1. Вступ

Постановка задачі:

- Є інформаційна система. Інформація з якої передається між клієнтами та системою у вигляді XML-документів. Необхідно забезпечити обробку цих документів.
- Обробка включає в себе дві задачі: аналіз вмісту документу (пошук інформації за ключовими словами та динамічну генерацію запитів) та трансформацію у файл HTML.
- Вхідні дані для аналізу та трансформації надаються у вигляді файлу-прикладу *.xml.
- Трансформація документу в HTML-код виконується на основі XSL-документа *.xsl.
- Аналіз вмісту документа повинен бути виконаний трьома способами – за допомогою SAX API, DOM API та LINQ to XML.
- Використання антишаблону магічна кнопка
- Використання шаблону «Стратегія» Strategy.
- Наявність діаграми класів

Варіант 6. "Успішність студентів":

Таблиці містять таку інформацію: П.І.П., факультет(департамент, відділення), кафедра, дані про навчальні дисципліни та успішність студентів та ін.

2. Принцип роботи програми

Після запуску програми на екрані з'являється «порожня» форма

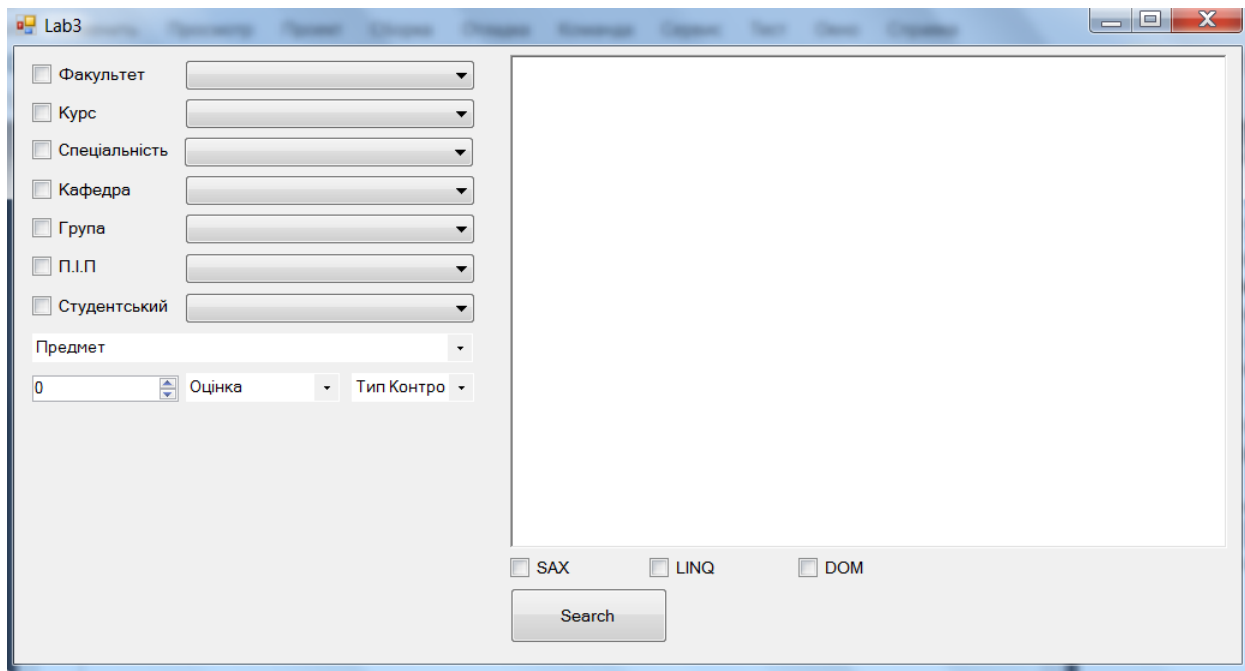


Рис. 2.1

Для створення форми використовувалися наступні елементи

- ComboBox (19)
- Button (2, кнопка з написом "Search" - антишаблон магічна кнопка (англ. *Magic pushbutton*))
- NumericUpDown (4)
- CheckBox (10)
- RichTextBox (1)
- FlowLayoutPanel (1)

На рис 2.2 зображено UML- діаграму класу Form1

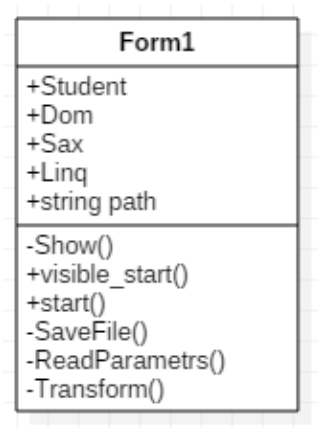


Рис. 2.2. Діаграма класу Form1

Принцип роботи:

1. За допомогою функції start() ([детальніше про XPath](#)) відбувається зчитування даних з файлу ([детальніше про xml-файл](#)) у ComboBox (зчитуються усі атрибути тегів) див. *рис. 2.4 -6*.

Факультет

Курс

Спеціальність

Кафедра

Рис. 2.3. До ComboBox додано атрибути усіх тегів <faculty>

Факультет

Курс

Спеціальність

Кафедра

Рис. 2.4. До ComboBox додано атрибути усіх тегів <course>

Факультет

Курс

Спеціальність

Кафедра

Група

П.І.П

Студентський

Предмет

Бази даних та інформаційні системи

Вступ до університетських студій

Генетика

ІУК

Математичний аналіз

Програмування

Теорія ймовірностей та математична статистика

Теорія функцій комплексної змінної

Хімія біоорганічна

Рис. 2.5. До ComboBox додано атрибути усіх тегів <subject>

2. За замовчуванням (функція visible_start()) відображається лише один блок комірок для введення критеріїв пошуку за назвою предмета, оцінками, типом контролю (рис. 2.7) решта стають «видимими» лише при заповненні хочаб одного з елементів попереднього блоку (рис. 2.8)

Предмет

0

Оцінка

Тип Контролю

Рис. 2.7.

Генетика

0

Оцінка

Тип Контролю

Предмет

0

Оцінка

Тип Контролю

Рис. 2.8.

3. Також за замовчуванням не відображається кнопка “HTML” (рис. 2.9) при натисканні, якої відбувається трансформація xml-файлу, що містить результат пошуку у html, за використанням *.xsl та *.css. Вона стає «видимим» після завершення аналізу даних (рис. 2.10)

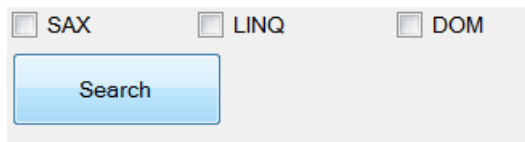


Рис. 2.9. Вигляд за замовчуванням

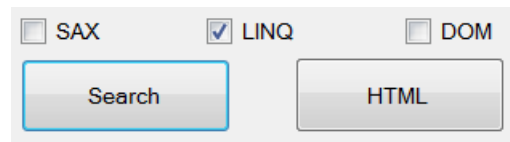


Рис. 2.10. Вигляд після аналізу даних

4. Заповнення ComboBox

Розглянемо ситуацію коли з усіх факультетів було обрано один («ННЦ Інститут біології»), далі ми розглядаємо лише ті курси, які «існують» на обраному факультеті, а не всі можливі курси (рис. 2.11), аналогічна ситуація із спеціальностями (рис. 2.12), студентами (рис. 2.13) та предметами (рис. 2.14).

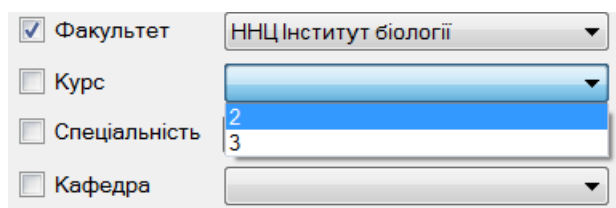


Рис. 2.11

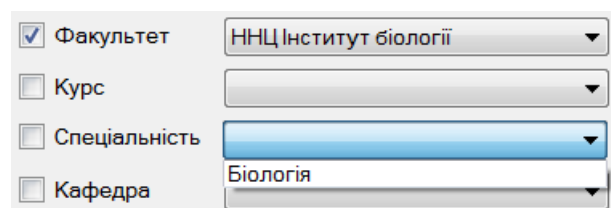


Рис. 2.12

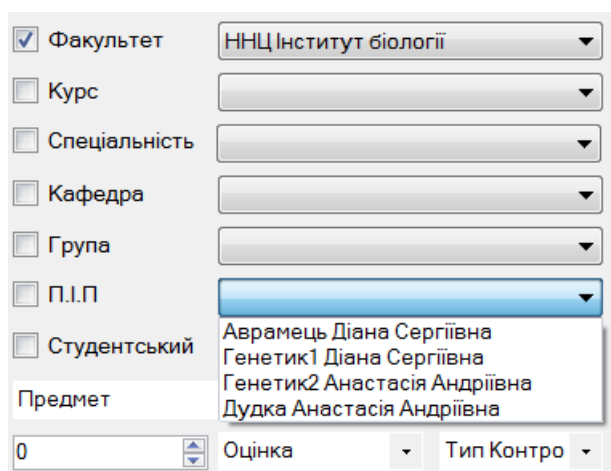


Рис. 2.13

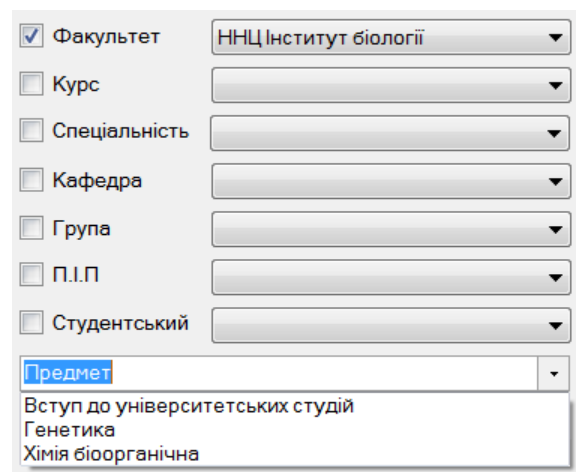


Рис. 2.14

5. Magic pushbutton

- При натисканні кнопки відбувається перевірка, який тип пошуку обрав користувач за допомогою CheckBox

Форму пошуку з фільтрами: Факультет (Факультет кібернетики), Курс (2), Спеціальність (Системний Аналіз), Група, П.І.П., Студентський. Нижче: Предмет, Оцінка (0), Тип Контролю. Вибрані методи пошуку: SAX, LINQ.

Рис. 2.12.

- якщо обрано більше ніж один тип (рис. 2.16), значення у RichTextBox виводиться повідомлення про необхідність обрання типу, а елементи CheckBox переходить у значення за замовчуванням (тобто *.Checked=false, де * імя CheckBox, що відповідають за тип пошуку)

Форму пошуку з повідомленням: 'Оберіть тип пошуку'. Методи пошуку не вибрані.

Рис. 2.16. Обрано більше ніж один тип аналізу даних

- якщо тип пошуку не було обрано, запускається DOM API (рис 2.17)

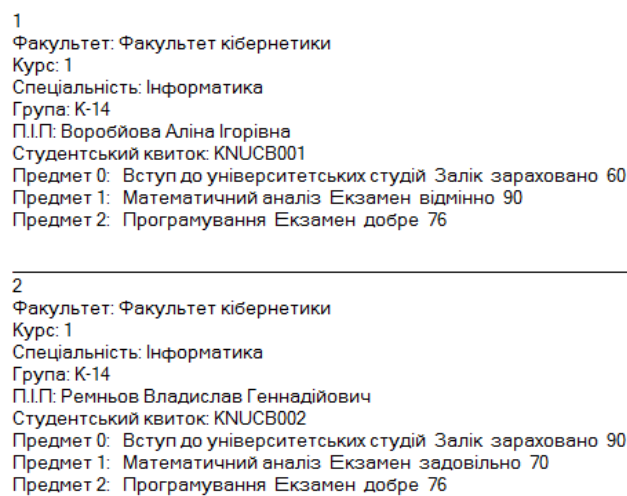
Форму пошуку з результатами пошуку за допомогою DOM API. Вибрано метод DOM. Результати пошуку: 1. Факультет: Факультет кібернетики, Курс: 2, Спеціальність: Системний Аналіз, Група: К-23, П.І.П. Запольський Станіслав Володимирович, Студентський квиток: KNUCB003, Предмет 0: Математичний аналіз Екзамен відмінно 90, Предмет 1: Програмування Екзамен добре 76.

Рис. 2.17. Тип пошуку не було обрано

6. Аналіз даних

- Після натискання магічної кнопки за допомогою функції `ReadParameters` відбувається зчитування даних з усіх видимих `ComboBox` і обирається один із способів аналізу даних.
- Детальніше про кожен спосіб:
 - 1) [LINQ](#)
 - 2) [SAX](#)
 - 3) [DOM](#)
- Детальніше про патерн `Strategy` та його використання [див. розділ 4](#)

7. Результати пошуку виводяться у `RichTextBox` (функція `Show()`)

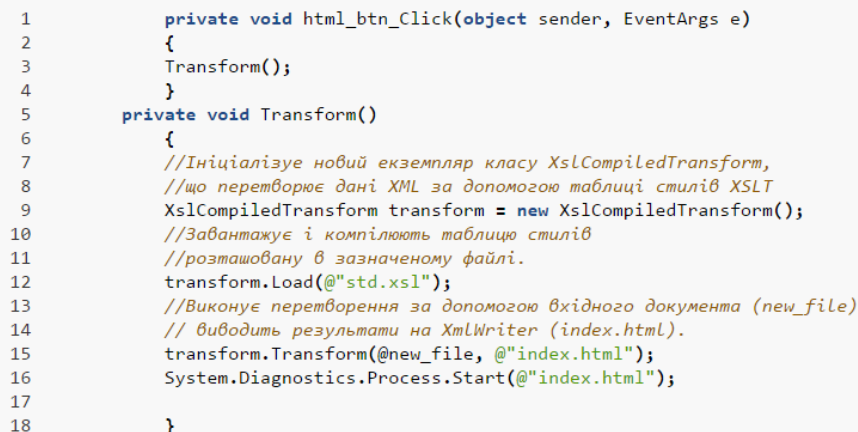


```
1
Факультет: Факультет кібернетики
Курс: 1
Спеціальність: Інформатика
Група: К-14
П.І.П: Воробйова Аліна Ігорівна
Студентський квиток: KNUCB001
Предмет 0: Вступ до університетських студій Залік зараховано 60
Предмет 1: Математичний аналіз Екзамен відмінно 90
Предмет 2: Програмування Екзамен добре 76

2
Факультет: Факультет кібернетики
Курс: 1
Спеціальність: Інформатика
Група: К-14
П.І.П: Ремньов Владислав Геннадійович
Студентський квиток: KNUCB002
Предмет 0: Вступ до університетських студій Залік зараховано 90
Предмет 1: Математичний аналіз Екзамен задовільно 70
Предмет 2: Програмування Екзамен добре 76
```

Рис. 2.13

8. Результати пошуку зберігаються у новий `xml`-файл (функція `SaveFile()`)
Детальніше [розділ 5](#)
9. Кнопка «HTML» стає «видимою», при її натисканні викликається функція `Transform()` (рис 2.19), що конвертує `xml`-файл у `html` з використанням `*.xsl` та `*.css`.



```
1 private void html_btn_Click(object sender, EventArgs e)
2 {
3     Transform();
4 }
5 private void Transform()
6 {
7     //Ініціалізує новий екземпляр класу XslCompiledTransform,
8     //що перетворює дані XML за допомогою таблиці стилів XSLT
9     XslCompiledTransform transform = new XslCompiledTransform();
10    //Завантажує і компілює таблицю стилів
11    //розташовану в зазначеному файлі.
12    transform.Load(@"std.xsl");
13    //Виконує перетворення за допомогою вхідного документа (new_file)
14    // виводить результати на XmlWriter (index.html).
15    transform.Transform(new_file, @"index.html");
16    System.Diagnostics.Process.Start(@"index.html");
17
18 }
```

Рис. 2.19

Детальніше про `xsl`-документ - [розділ 6](#), `html` - [розділ 7](#), `css` - [розділ 8](#)

3. Структура XML-файлу

За допомогою Notepad++ було створено xml-файл, за наступним принципом:

- `<?xml version="1.0" encoding="utf-8" ?>` - xml-декларація, що складається з таких елементів:
 - номер версії, `<? xml version = "1.0" ... ?>` - це обов'язковий параметр. Номер в майбутніх версіях XML може змінитися. Поточна версія, яка використовується в даному файлі - 1.0.
 - оголошення кодування `<? ... encoding = " utf-8"? >` - це необов'язковий параметр. Якщо він використовується, то оголошення кодування повинно розташовуватися відразу після інформації про версію.
- `<studentDataBase>` - єдиний кореневий елемент, в якому містяться вкладені теги `<faculty>`.

Атрибути яких є назви факультетів КНУ ім. Т.Г.Шевченка або якогось іншого реального або вигаданого внз. Кожен факультет має курс (`<course>`), кожен курс має спеціальність(`<speciality>`), спеціальність-кафедру(`<department>`), кафедра – групу(`<group>`).

Відповідно кожна група складається з студентів(`<student>`). Кожен студент має ім'я (атрибут "NAME") та номер студентського квитка (атрибут "IDCARD") та декілька (мінімум 1, інакше програма відмовиться працювати з даним файлом, максимум 4, інакше обробляються лише перші 4 предмети) вкладених тегів `<subject>`, що мають атрибути:

- "SUB" – назва предмету, який складав даний студент
- "TESTYPE" – форма перевірки знань з даного предмету (залік або екзамен, інші форми контролю не розглядаються)
- "MARK" – оцінка за 100-бальною шкалою (мінімум 60, максимум 100, інакше програма відмовиться працювати з даним файлом)
- "TYPEMARK" - оцінка за «національною шкалою» (відмінно, добре, задовільно та зараховано для заліків)

Варіанти, що студента не було допущено до складання, відправлено на перездачу та інше не розглядаються, припустимо, що таких студентів не існує.

Кожен тег `<faculty>`, `<course>`, `<speciality>`, `<group>` має містити атрибут ("FNAME", "COURSE", "SPEC", "GROUP"), що відповідає назві або номеру даного факультету, курсу і т.д.

Тег `<department>` може не містити атрибутів, якщо атрибут тегу `<course>` має значення менше 3. Тобто студенти, що навчаються на 1-2 курсах не обирають кафедру.

Структура xml-файлу:

```
<studentsDataBase>
  <faculty FNAME="Факультет кібернетики">
    <course COURSE="1">
      <speciality SPEC="Інформатика">
        <department>
          <group GROUP="К-14">
            <student NAME="Воробйова Аліна Ігорівна" IDCARD="KNUCB001">
              <subject SUB="Вступ до університетських студій" TESTTYPE="Залік" MARK="60" TYPEMARK="зараховано"></subject>
              <subject SUB="Математичний аналіз" TESTTYPE="Екзамен" MARK="90" TYPEMARK="відмінно"></subject>
              <subject SUB="Програмування" TESTTYPE="Екзамен" MARK="76" TYPEMARK="добре"></subject>
            </student>
            <student NAME="Ремньов Владислав Геннадійович" IDCARD="KNUCB002">
              <subject SUB="Вступ до університетських студій" TESTTYPE="Залік" MARK="90" TYPEMARK="зараховано"></subject>
              <subject SUB="Математичний аналіз" TESTTYPE="Екзамен" MARK="70" TYPEMARK="задовільно"></subject>
              <subject SUB="Програмування" TESTTYPE="Екзамен" MARK="76" TYPEMARK="добре"></subject>
            </student>
          </group>
          <group GROUP="К-15">...</group>
        </department>
      </speciality>
    </course>
    <course COURSE="3">...</course>
    <course COURSE="2">...</course>
  </faculty>
  <faculty FNAME="ІНЦ Інститут біології">...</faculty>
</studentsDataBase>
```

Рис. 3.1. Структура xml-файлу

4. Паттерн Стратегія

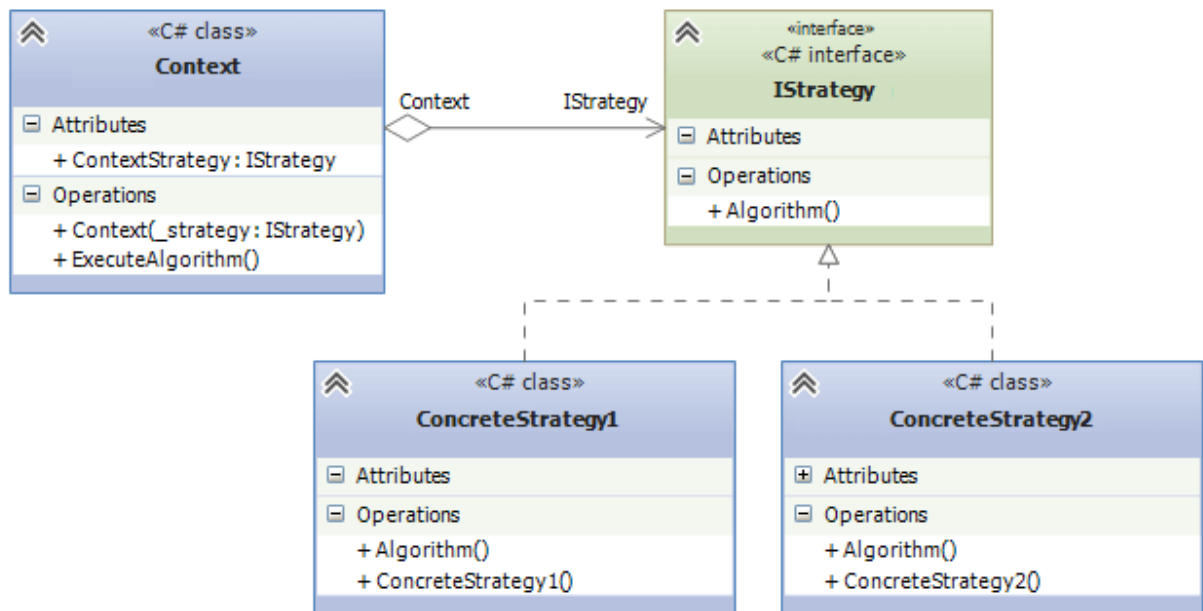


Рис 4.1.Strategy

Стратегія (Strategy) — шаблон проектування, належить до класу шаблонів поведінки.

Це шаблон проектування, який визначає набір алгоритмів, інкапсулює кожен з них і забезпечує їх взаємозамінність. Залежно від ситуації ми можемо легко замінити один використовуваний алгоритм іншим. При цьому заміна алгоритму відбувається незалежно від об'єкта, який використовує даний алгоритм.

Коли використовувати стратегію?

- Коли є кілька споріднених класів, які відрізняються поведінкою. Можна задати один основний клас, а різні варіанти поведінки винести в окремі класи і при необхідності їх застосовувати
- Коли необхідно забезпечити вибір з кількох варіантів алгоритмів, які можна легко міняти в залежності від умов
- Коли необхідно змінювати поведінку об'єктів на стадії виконання програми
- Коли клас, який застосовує певну функціональність, нічого не повинен знати про її реалізацію

Переваги патерну Strategy

- + Систему простіше підтримувати і модифікувати, так як сімейство алгоритмів перенесено в окрему ієрархію класів.
- + Патерн Strategy надає можливість заміни одного алгоритму іншим в процесі виконання програми.
- + Патерн Strategy дозволяє приховати деталі реалізації алгоритмів від клієнта.

Недоліки патерну Strategy

- Для правильного налаштування системи користувач повинен знати про особливості всіх алгоритмів.
- Число класів в системі, побудованої із застосуванням патерну Strategy, зростає.

Застосування патерну «Стратегія» в умовах даної лабораторної роботи

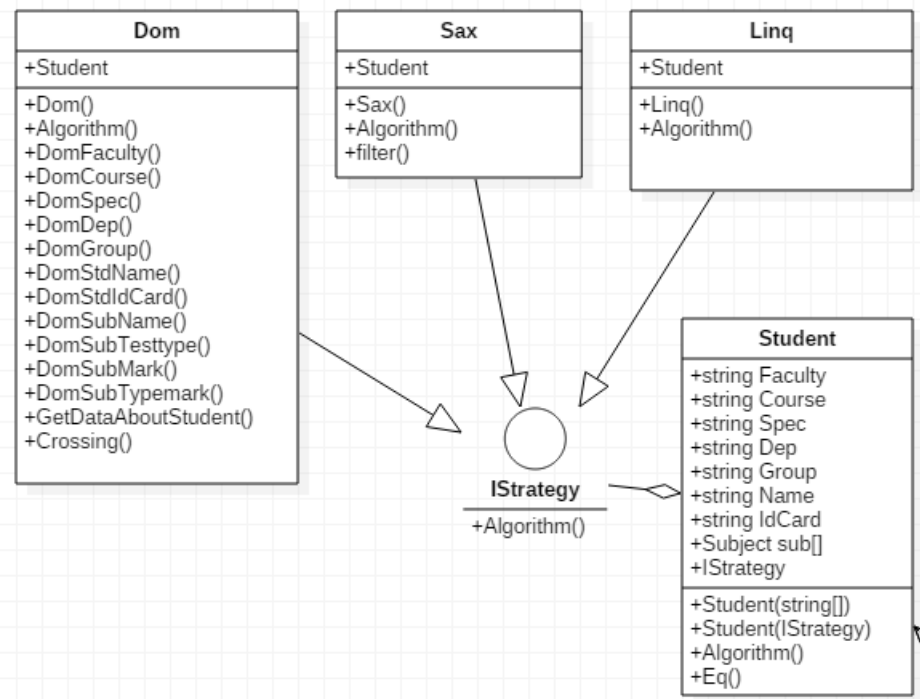


Рис 4.2

Як видно з діаграми (рис 4.2), тут є такі учасники:

- Інтерфейс IStrategy, який визначає метод Algorithm (). Це спільний інтерфейс для всіх реалізуючих його алгоритмів. Замість інтерфейсу тут також можна було б використовувати абстрактний клас.
- Класи Dom, Sax і Linq, які реалізують інтерфейс IStrategy, надаючи свою версію методу Algorithm ().
- Клас Student зберігає посилання на об'єкт IStrategy і пов'язаний з інтерфейсом IStrategy відношенням агрегації.

4.1 LINQ

LINQ (Language-Integrated Query) представляє простий і зручний мову запитів до джерела даних. Як джерело даних може виступати об'єкт, який реалізує інтерфейс `IEnumerable` (наприклад, стандартні колекції, масиви), набір даних `DataSet`, документ XML. Але незалежно від типу джерела LINQ дозволяє застосувати до всіх один і той же підхід для вибірки даних.

Існує кілька різновидів LINQ:

1. LINQ to Objects: застосовується для роботи з масивами і колекціями
2. LINQ to Entities: використовується при зверненні до баз даних через технологію Entity Framework
3. LINQ to Sql: технологія доступу до даних в MS SQL Server
4. LINQ to XML: застосовується при роботі з файлами XML
5. LINQ to DataSet: застосовується при роботі з об'єктом `DataSet`
6. Parallel LINQ (PLINQ): використовується для виконання паралельної запитів

Для реалізації пошуку за допомогою LINQ. Створюємо наступний клас:

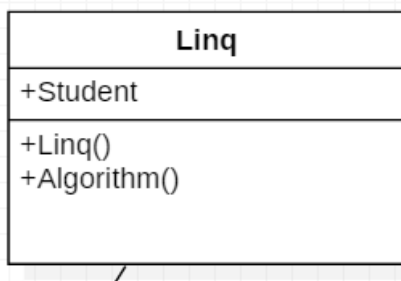


Рис 4.1.1

- `Linq(path)` – конструктор, що за допомогою методу `Load()` створює новий `XDocument` з файлу. `Path` - рядок, що посилається на файл для завантаження в новий `XDocument`
- `Algorithm()`
- 1. За допомогою мови `Linq` відбувається фільтрація за усіма параметрами пошуку одночасно
- Запит має таку структуру:

```
1 var result = (from val in doc.Descendants("student")
2               where
3               (
4                 student.Name == null || student.Name == val.Attribute("NAME").Value
5               )
6               select val).ToList();
```

Рис 4.1.2

- Запит починається з конструкції «*from змінна in набір об'єктів*» в даному випадку множиною об'єктів є колекція вузлів, що повертає метод `XContainer.Descendants(XName)`, тільки елементи, які мають відповідний `XName`, включаються в колекцію. (тобто усі елементи “student”).

- Для вибору елементів з деякого набору за умовою використовується метод `where`. Якщо вираз в методі `where` для певного елемента дорівнюватиме `true` (якщо параметр пошуку за іменем не задано, або атрибут конкретного “NAME” студента рівний заданому параметру пошуку), то даний елемент потрапляє в результуючу вибірку.

```
1 var result = (from val in doc.Descendants("student")
2   where
3   (
4     (student.Spec == null || student.Spec == ((val.Parent).Parent).Parent.Attribute("SPEC").Value)
5     &&
6     (student.Name == null || student.Name == val.Attribute("NAME").Value)
7   )
8   select val).ToList();
```

Рис 4.1.3

- На *рис 4.1.3* було збільшено кількість фільтрів, тепер елемент потрапляє у вибірку, якщо знайдено шуканого студента з конкретним іменем та спеціальністю, або параметрів пошуку за цими критеріями не було задано.

```
1 var result = (from val in doc.Descendants("student")
2   where
3   (
4
5     (student.Name == null || student.Name == val.Attribute("NAME").Value) &&
6     (val.Descendants("subject").Any(element => (
7       (student.sub.Name == "" || student.sub.Name == element.Attribute("SUB").Value)&&
8       (student.sub.Testtype == "" || student.sub.Testtype == element.Attribute("TESTTYPE").Value) &&
9       (student.sub.Mark == "" || student.sub.Mark == element.Attribute("MARK").Value) &&
10      (student.sub.TypeMark == "" || student.sub.TypeMark == element.Attribute("TYPEMARK").Value)
11    )
12   select val).ToList();
```

Рис 4.1.4

- Фільтр за предметом і іменем має таку структуру *рис 4.1.4*
 - Тут використовується метод `Any()`, що визначає, чи задовольняє хоча б один елемент колекції даних умові.
 - Усі інші фільтри мають абсолютно аналогічну структуру, як фільтри на *рис 4.1.3*
2. Збереження даних знайденого студента.
- Після того, як було знайдено студента, що відповідає усім критеріям пошуку, усі його дані додаються до списку «знайдених студентів», після завершення опрацювання файлу метод `Algorithm()` повертає цей список.

4.2 DOM

Об'єктна модель документа (англ. Document Object Model, DOM) — специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами (як правило, документів XML). Визначається ця специфікація консорціумом W3C.

З точки зору об'єктно-орієнтованого програмування, DOM визначає класи, методи та атрибути цих методів для аналізу структури документів та роботи із представленням документів у вигляді дерева. Все це призначено для того, аби надати можливість комп'ютерній програмі доступу та динамічної модифікації структури, змісту та оформлення документа.

Для реалізації пошуку за допомогою DOM API. Створюємо наступний клас:

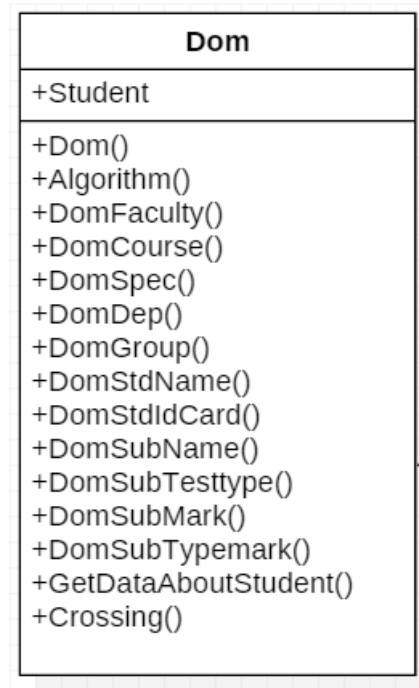


Рис 4.2.1 Клас Dom

- Dom (path) – конструктор, що за допомогою методу Load() створює новий XmlDocument з файлу. Path - рядок, що посилається на файл для завантаження в новий XmlDocument
- Оскільки маємо дерево-подібну структуру, то щоб знайти шуканого студента необхідно обійти все дерево. Використовуючи рекурсивний виклик методу Algorithm() будемо рухатись по вузлах дерева.



Рис 4.2.2 Модель DOM

На *рис 4.2.2* показано частину, яка буде створена в пам'яті, якщо зчитати даний *рис 3.1* xml-файл в модель структури DOM.

Кожне коло в даній ілюстрації є вузол в структурі XML-документа (об'єкт XmlNode). Об'єкт XmlNode є базовим об'єктом дерева DOM. Клас XmlDocument, що розширює клас XmlNode, підтримує методи для виконання операцій над документом в цілому (наприклад, завантаження його в пам'ять або збереження XML в файл). Крім того, XmlDocument надає можливості для перегляду вузлів всього XML-документа і виконання операцій над ними.

Многокутники зображають атрибути та їх значення. Атрибути не є вузлами, що містяться в батьківських, дочірніх і однорівневих зв'язках. Атрибути вважаються власністю вузла елемента і являють собою пару «ім'я-значення».

4.3 SAX

Стандартним засобом для читання і маніпулювання XML-файлами є Об'єктна Модель Документа, Document Object Model (DOM). На жаль, цей метод, який включає в себе читання всього файлу і збереження його в структурі дерева, може бути малоефективним, повільним і вимагати багато ресурсів.

Однією з альтернатив йому є Simple API for XML або SAX. SAX дозволяє обробляти документ у міру його читання, що усуває необхідність очікувати з виконанням якихось дій, поки весь документ не буде збережений.

Для реалізації пошуку за допомогою SAX. Створюємо наступний клас:

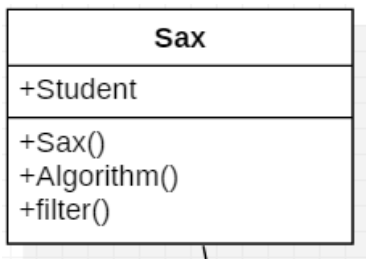


Рис 4.3.1

- Sax (path) – конструктор, що ініціалізує новий екземпляр класу XmlTextReader зазначеним файлом. Path - рядок , що посилається на xml-файл.
- Algorithm() – повертає список студентів, що відповідає критеріям пошуку.
Алгоритм пошуку:

Файл читається, поки не знайдено тег <faculty>(XmlTextReader.Read()),

Якщо атрибут «FNAME» відповідає назві шуканого факультету, або параметр пошуку за цим критерієм не було задано, значення атрибуту зберігається як факультет, де навчається студент

Файл читається , поки не знайдено тег <course>

Якщо атрибут «COURSE» відповідає номеру шуканого курсу, або параметр пошуку за цим критерієм не було задано, значення атрибуту зберігається, як курс на якому навчається студент

.....

Файл читається , поки не знайдено тег <subject> значення усіх атрибутів усіх тегів <subject>, викликається функція filter(), яка повертає значення true, якщо необхідна кількість тегів <subject> містить шуканий набір значень аргументів або параметрів пошуку за назвами предметів, оцінками та критеріями оцінювання не було задано.

Якщо filter() повертає значення true, до списку додається новий студент (який містить усі зчитані дані). Файл читається, поки не знайдено наступний відкриваючий тег <faculty>, або відкриваючий тег <course> , або або відкриваючий тег <student>, або кінець файлу.

Інакше

Файл читається, поки не знайдено наступний відкриваючий тег <faculty> або відкриваючий тег <course> або кінець файлу.

Інакше

Файл читається, поки не знайдено наступний відкриваючий тег <faculty> або кінець файлу.

4.4 Як вибрати між SAX і DOM

Використовувати DOM або SAX, залежить від кількох факторів:

- Призначення програми: Якщо ви збираєтеся робити зміни в даних і виводити їх як XML, то в більшості випадків способом для цього є DOM. Не можна сказати, що ви не можете робити зміни при використанні SAX, але цей процес значно складніший, так як ви повинні робити зміни в копії даних, а не в самих даних.
- Обсяг даних: Для великих файлів SAX є найкращим вибором.
- Як будуть використані дані: Якщо насправді буде використана тільки невелика частина даних, вам, можливо, краще застосувати SAX, щоб виділити її в вашу програму. З іншого боку, якщо ви знаєте, що вам доведеться посылатися назад у великому обсязі інформації, яка вже була оброблена, SAX, можливо, не є правильним вибором.
- Вимоги до швидкодії: реалізації SAX зазвичай швидше, ніж реалізації DOM.
- Важливо пам'ятати, що SAX і DOM не є взаємовиключними. Ви можете використовувати DOM для створення потоку подій SAX, і ви можете використовувати SAX для створення дерева DOM. Фактично, більшість парсерів, застосовуваних для створення дерева DOM, використовують SAX, щоб зробити це!

5. Збереження результату аналізу даних

Результати пошуку зберігаються у новий xml-файл, що створюється використовуючи функціональність LINQ to XML у методі SaveFile () (рис 5.1).

```
1 private void SaveFile(List<Student> inform, string path)
2 {
3     XDocument xdoc = new XDocument(new XDeclaration("1.0", "utf-8", "yes"));
4
5     XElement studentsDataBase = new XElement("studentsDataBase");
6
7     foreach (Student inf in inform)
8     {
9         XElement faculty = new XElement("faculty");
10         XAttribute facultyAttr = new XAttribute("FNAME", inf.Faculty);
11         XElement course = new XElement("course");
12         XAttribute courseAttr = new XAttribute("COURSE", inf.Course);
13         XElement speciality = new XElement("speciality");
14         XAttribute specialityAttr = new XAttribute("SPEC", inf.Spec);
15         XElement department = new XElement("department");
16         XAttribute departmentAttr = new XAttribute("DEP", "пусто");
17         if (inf.Dep != "") { departmentAttr = new XAttribute("DEP", inf.Dep); }
18         XElement group = new XElement("group");
19         XAttribute groupAttr = new XAttribute("GROUP", inf.Group);
20         XElement student = new XElement("student");
21         XAttribute studentName = new XAttribute("NAME", inf.Name);
22         XAttribute studentIdCard = new XAttribute("IDCARD", inf.IdCard);
23
24         XElement[] subject = new XElement[count];
25         XAttribute[] subjectName = new XAttribute[count];
26         XAttribute[] subjectTesttype = new XAttribute[count];
27         XAttribute[] subjectMark = new XAttribute[count];
28         XAttribute[] subjectTypemark = new XAttribute[count];
29         for (int i = 0; i < subject.Count(); i++)
30         {
31             if (inf.sub[i].Name == "") break;
32             subject[i] = new XElement("subject");
33             subjectName[i] = new XAttribute("SUB", inf.sub[i].Name);
34             subjectTesttype[i] = new XAttribute("TESTTYPE", inf.sub[i].Testtype);
35             subjectMark[i] = new XAttribute("MARK", inf.sub[i].Mark);
36             subjectTypemark[i] = new XAttribute("TYPEMARK", inf.sub[i].Typemark);
37         }
38         for (int i = 0; i < subject.Count(); i++)
39         {
40             if (inf.sub[i].Name == "") break;
41             subject[i].Add(subjectName[i]);
42             subject[i].Add(subjectTesttype[i]);
43             subject[i].Add(subjectMark[i]);
44             subject[i].Add(subjectTypemark[i]);
45             student.Add(subject[i]);
46         }
47
48         student.Add(studentName);
49         student.Add(studentIdCard);
50
51         group.Add(student);
52         group.Add(groupAttr);
53
54         department.Add(group);
55         if (inf.Dep != "")
56             department.Add(departmentAttr);
57
58         speciality.Add(department);
59         speciality.Add(specialityAttr);
60
61         course.Add(speciality);
62         course.Add(courseAttr);
63
64         faculty.Add(course);
65         faculty.Add(facultyAttr);
66
67         studentsDataBase.Add(faculty);
68     }
69
70     xdoc.Add(studentsDataBase);
71     xdoc.Save(path);
```

Рис. 5.1 SaveFile()

Загальні класи простору імен:

- XAttribute: представляє атрибут xml-елемента
- XComment: подає коментар
- XDocument: представляє весь xml-документ
- XElement: представляє окремий xml-елемент

Щоб створити документ, потрібно створити об'єкт класу XDocument. Це об'єкт верхнього рівня в XML-документі.

Елементи створюються за допомогою конструктора класу XElement. Параметр конструктора передає назву елемента, наприклад, faculty. Атрибут створюється аналогічно.

Потім атрибути і вкладені елементи додаються в елементи, яким вони мають належати за допомогою методу Add ().

Так як документ xml повинен мати один кореневий елемент, то потім все елементи додаються в один контейнер - елемент studentsDataBase.

В кінці кореневий елемент додається в об'єкт XDocument, і цей об'єкт зберігається на диску в xml-файл за допомогою методу Save (path). Path - рядок, що посилається на xml-файл.

Якщо відкриємо збережений файл у вікні браузеру, то побачимо в ньому такий зміст:

```
▼<studentsDataBase>
  ▼<faculty FNAME="Факультет кібернетики">
    ▼<course COURSE="3">
      ▼<speciality SPEC="Прикладна математика">
        ▼<department DEP="Обчислювальна математика">
          ▼<group GROUP="ОМ-3">
            ▼<student NAME="Прокопєць Вадим Георгійович" IDCARD="KNUCB005">
              <subject SUB="Теорія ймовірностей та математична статистика" TESTTYPE="Екзамен" MARK="89" TYPEMARK="зараховано"/>
              <subject SUB="Бази даних та інформаційні системи" TESTTYPE="Екзамен" MARK="90" TYPEMARK="відмінно"/>
              <subject SUB="Теорія функцій комплексної змінної" TESTTYPE="Екзамен" MARK="76" TYPEMARK="добре"/>
            </student>
          </group>
        </department>
      </speciality>
    </course>
  </faculty>
  ▼<faculty FNAME="Факультет кібернетики">
    ▼<course COURSE="3">
      ▼<speciality SPEC="Інформатика">
        ▼<department DEP="ТТП">
          ▼<group GROUP="ТТП-3">
            ►<student NAME="Яшук Софія Юріївна" IDCARD="KNUCB005">...</student>
          </group>
        </department>
      </speciality>
    </course>
  </faculty>
</studentsDataBase>
```

Рис 5.2 Структура xml-файлу

6. XSL

Extensible Stylesheet Language — сімейство рекомендацій для визначення перетворень і представлення XML документів. XSL складається із трьох частин:

XSL Transformations (XSLT) - мова для описання перетворень XML документів.

XML Path Language (XPath) - мова для виразів, використовується XSLT для доступу або посилання на частини XML документа. (XPath також використовується в специфікації XML Linking).

XSL Formatting Objects (XSL-FO) - XML словник для описання семантики форматування.

6.1 XPath

XPath представляє мову запитів в XML. Він дозволяє вибирати елементи, що відповідають певному селектору.

Приклад 1

Цикл обирає з файлу усі теги <faculty>, що мають атрибут “FNAME” та додає до Combobox

```
1 foreach (XmlNode Name in doc.SelectNodes("//faculty[@FNAME]"))
2     {
3         faculty_combo.Items.Add(Name.Attributes["FNAME"].Value);
4     }
5
```

Рис 6.1.1

Приклад 2

В залежності від того чи було обрано критерії пошуку за курсом та факультетом формується рядок XPath. За цим шляхом в циклі обираються теги <speciality>, що мають атрибут “SPEC” та додає до Combobox, уникаючи повторень.

```
1 if (Convert.ToString(faculty_combo.SelectedItem) != "")
2     {
3         road = "//faculty[@FNAME=\"" + Convert.ToString(faculty_combo.SelectedItem) +
4             "\" ]/course[@COURSE=\"" + Convert.ToString(course_combo.SelectedItem) +
5             "\" ]/speciality";
6     }
7     else
8     {
9         road = "//course[@COURSE=\"" + Convert.ToString(course_combo.SelectedItem) +
10             "\" ]/speciality";
11     }
12
13     foreach (XmlNode val in doc.SelectNodes(road))
14     {
15         bool check = true;
16         foreach (var Item in speciality_combo.Items)
17         {
18             if (Convert.ToString(Item) == Convert.ToString(val.Attributes["SPEC"].Value))
19                 check = false;
20         }
21         if (check) speciality_combo.Items.Add(val.Attributes["SPEC"].Value);
22     }
```

Рис 6.1.2

6.2 XSLT

За допомогою XSLT можна трансформувати XML-документ в будь-який вид, будь то HTML, WML, RTF, PDF, SQL, SWF. XSL несе в собі інформацію про те, як буде оформлений документ, де і як повинні розташовуватися дані.

В даному випадку відбувається трансформація у XML.

Структура xsl-документу:

```
1 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
2 <xsl:output method="html" indent="yes"/>
3 <xsl:template match="studentsDataBase">
4 <HTML>
5 <HEAD>
6 <link rel="stylesheet" href="style.css" type="text/css"/>
7 </HEAD>
8 <BODY>
9 <!-- <xsl:apply-templates/> -->
10 <article>
11 <xsl:for-each select="faculty">
12 <table border="1">
13 <tr>
14 <caption>
15 <xsl:value-of select="@FNAME"/>
16 </caption>
17 </tr>
18 <xsl:for-each select="course">
19 <tr>
20 <th>Курс</th>
21 <th>
22 <xsl:value-of select="@COURSE"/>
23 </th>
24 </tr>
25 <xsl:for-each select="speciality">
26 <tr>
27 <th>Спеціальність</th>
28 <th>
```

```
48 <th>Студент</th>
49 <th>
50 <xsl:value-of select="@NAME"/>
51 </th>
52 </tr>
53 <tr>
54 <th>ID</th>
55 <th>
56 <xsl:value-of select="@IDCARD"/>
57 </th>
58 </tr>
59 <table>
60 <tr>
61 <td>Предмет</td>
62 <td>Тип контролю</td>
63 <td>Оцінка/100</td>
64 <td>Оцінка</td>
65 </tr>
66 <xsl:for-each select="subject">
67 <tr>...</tr>
68 </xsl:for-each>
69 </table>
70 </xsl:for-each>
```

Рис. 6.2.1

Елементи xsl:

- Перший рядок файлу містить тег елемента `xsl: stylesheet`. Атрибути елемента - номер версії і посилання на простір імен. Ці атрибути елемента `xsl: stylesheet` є обов'язковими. У нашому випадку деталі про використаний простір імен міститься за посиланням <http://www.w3.org/1999/XSL/Transform>.
 - `version` - обов'язковий атрибут, в якому вказується версія мови XSLT, використана при створенні цього перетворення.
- Елемент верхнього рівня `xsl: output` дозволяє вказувати, яким чином має бути виведено результуюче дерево.
 - `method` - необов'язковий атрибут, який визначає, який метод повинен використовуватися для виведення документа. Значення цього атрибута може бути будь-яке ім'я, але при цьому технічна рекомендація XSLT визначає тільки три стандартних методу виведення - "xml", "html" і "text". В данному випадку використовується "html".
 - `indent` вказує, чи може XSLT процесор ставити додаткові пробіли при виведенні результату, атрибут повинен мати значення `yes` або `no`.
- Елемент верхнього рівня `xsl: template` визначає повторно використовуваний шаблон для формування бажаного виведення для вузлів певного типу в певному контексті
 - `match` - необов'язковий атрибут, зразок вузлів дерева, для перетворення яких слід застосовувати даний шаблон.
- `xsl:for-each` Застосовує шаблон кілька разів - по черзі до кожного вузла множини.
 - `select` - обов'язковий атрибут, вказує вираз, результатом обчислення якого має бути множина вузлів. Шаблон, що міститься в `xsl: for-each`, буде виконаний процесором для кожного вузла цієї множини.
- `xsl:value-of` вставляє значення обраного вузла у вигляді тексту
 - `select` - обов'язковий атрибут, задає вираз, яке обчислюється процесором, потім перетворюється в рядок і виводиться у вигляді тексту.

7. HTML

Розглянемо структуру html- файлу на прикладі:

```
1 <HTML>
2     <HEAD>
3         <META http-equiv="Content-Type" content="text/html" charset="utf-8">
4         <link rel="stylesheet" href="style.css" type="text/css">
5     </HEAD>
6     <BODY>
7     <article>
8     <table>
9     <tr>
10        <caption>Факультет кібернетики</caption>
11    </tr>
12        <th>Студент</th>
13        <th>Воробйова Аліна Ігорівна</th>
14    </tr>
15    <table>
16    <tr>
17        <td>Предмет</td>
18        <td>Тип контролю</td>
19        <td>Оцінка/100</td>
20        <td>Оцінка</td>
21    </tr>
22    <tr>
23        <td>Вступ до університетських студій</td>
24        <td>Залік</td>
25        <td>60</td>
26        <td>зараховано</td>
27    </tr>
28    </table>
29    </table>
30        <br>
31    </article>
32 </BODY>
33 </HTML>
```

Рис 7.1. Приклад html-файл

| Факультет кібернетики | | | |
|----------------------------------|--------------------------|------------|------------|
| Студент | Воробйова Аліна Ігорівна | | |
| Предмет | Тип контролю | Оцінка/100 | Оцінка |
| Вступ до університетських студій | Залік | 60 | зараховано |

Рис 7.2. Вигляд прикладу у вікні браузера

Тег <head> призначений для зберігання інших елементів, мета яких - допомогти браузеру в роботі з даними. Також всередині контейнера <head> знаходяться метатеги, які використовуються для зберігання інформації призначеною для браузерів і пошукових систем.

- <meta> визначає метатеги, які використовуються для зберігання інформації призначеною для браузерів і пошукових систем.
 - http-equiv - призначений для конвертації метатега в заголовок HTTP.
 - content - встановлює значення атрибута, заданого за допомогою name або http-equiv.
 - charset - задає кодування документа.

- `<link>` встановлює зв'язок із зовнішнім документом на зразок файлу зі стилями або з шрифтами. На відміну від тега `<a>`, тег `<link>` розміщується завжди всередині контейнера `<head>` і не створює посилання.
 - `href` - шлях до файлу.
 - `rel` - визначає відносини між поточним документом і файлом, на який робиться посилання.
 - `type` - повідомляє браузеру, який MIME*-тип даних використовується для зовнішнього документа. Як правило, застосовується для того, щоб вказати, що підключається файл містить CSS. Для підключення таблиць стилів застосовується тип `text / css`.
- MIME (Multipurpose Internet Mail Extension, Багатоцільові розширення пошти Інтернету) - специфікація для передачі по мережі файлів різного типу: зображень, музики, текстів, відео, архівів та ін. Вказівка MIME-типу використовується в HTML зазвичай при передачі даних форм і вставки на сторінку різних об'єктів.

Елемент `<body>` призначений для зберігання змісту веб-сторінки (контенту), що відображається у вікні браузера.

- Тег `<article>` задає вміст сайту на кшталт новини, статті, записи блогу, форуму або ін.
- Елемент `<table>` служить контейнером для елементів, що визначають вміст таблиці. Будь-яка таблиця складається з рядків і комірок, які задаються за допомогою тегів `<tr>` і `<td>`. Усередині `<table>` допустимо використовувати наступні елементи: `<caption>`, `<col>`, `<colgroup>`, `<tbody>`, `<td>`, `<tfoot>`, `<th>`, `<thead>` і `<tr>`.
- `<tr>` служить контейнером для створення рядки таблиці. Кожна клітинка в межах такого рядка може задаватися за допомогою тега `<th>` або `<td>`.
- `<caption>` призначений для створення заголовка до таблиці і може розміщуватися тільки всередині контейнера `<table>`, причому відразу після відкриваючого тега. Такий заголовок являє собою текст, за замовчуванням відображається перед таблицею.
- `
` перенесення на новий рядок

Так виглядатиме результат трансформації у вікні браузеру

Факультет кібернетики

| | | | |
|---------------------|--------------|-------------------------------------|----------|
| Курс | | 2 | |
| Спеціальність | | Системний Аналіз | |
| Кафедра | | | |
| Група | | К-23 | |
| Студент | | Запольський Станіслав Володимирович | |
| ID | | KNUCBoo3 | |
| Предмет | Тип контролю | Оцінка/100 | Оцінка |
| Математичний аналіз | Екзамен | 90 | відмінно |
| Програмування | Екзамен | 76 | добре |

Рис 7.3

8. CSS

Каскадні таблиці стилів (CSS) — спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних на HTML та XHTML.

В даній лабораторній роботі використовувалися зовнішні таблиці стилів (окремий файл *.css, який підключається до створеного при трансформації html за допомогою тегу link).

Використовуваний css-файл:

```
1  body
2  {
3      outline:0;
4      margin:0;
5      padding:0;
6      font:25px Georgia;
7      color:#000;
8      width:100%;
9      height:100%;
10 }
11
12 article
13 {
14     margin: 20px auto auto;
15     padding:40px;
16     width:840px;
17     height:100%;
18 }
19
20 table
21 {
22     border-collapse: collapse;
23     width: 800px;
24     table-layout: fixed;
25     border: 0px;
26     text-align: left;
27 }
28
29 caption
30 {
31     font-size:28px;
32     font-weight:bold;
33 }
34
35 tr
36 {
37     .....
38     background:#F5F5F5;
39     border-top: 1px dashed #fff;
40 }
41
42 tr:hover
43 {
44     background: #800000;
45     color: #ffffff;
46 }
47
48 tr:nth-child(2n)
49 {
50     background:#DEDEE5;
51 }
52
53 td
54 {
55     .....
56     height:15px;
57     padding: 8px;
58     border:0px;
59 }
60
61
62
63
64
65
66
```

Рис 8.1 CSS-файл

Структура css-правила:

Будь-яке CSS правило складається з двох частин: CSS селектор, за допомогою селекторів задаються елементи, до яких застосовуються CSS правила і блоки оголошень, блок оголошень може складатися з одного або декількох CSS оголошень. У свою чергу, кожне оголошення складається з двох частин: CSS властивість і значення. Таким чином каскадна таблиця стилів CSS складається з набору CSS правил.

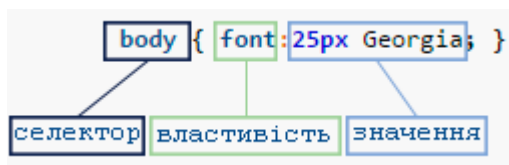


Рис 8.2 Приклад CSS правила

На *рис 8.2* видно, що всі стилі, які необхідно задати для певного HTML елемента або групи елементів задаються у фігурних дужках, перед якими пишеться селектор. У середині фігурних дужок розміщуються оголошення: пара властивість і значення, між собою вони розділяються двокрапкою, після кожного оголошення ставиться крапка з комою. В даному файлі (*рис 8.1*) задаються стилі для таких елементів:

body, article, table, th, tr, td, caption.

У оголошеннях використовуються наступні властивості:

- **margin** - Встановлює величину відступу від кожного краю елемента. Відступом є простір від кордону поточного елемента до внутрішньої межі його батьківського елемента

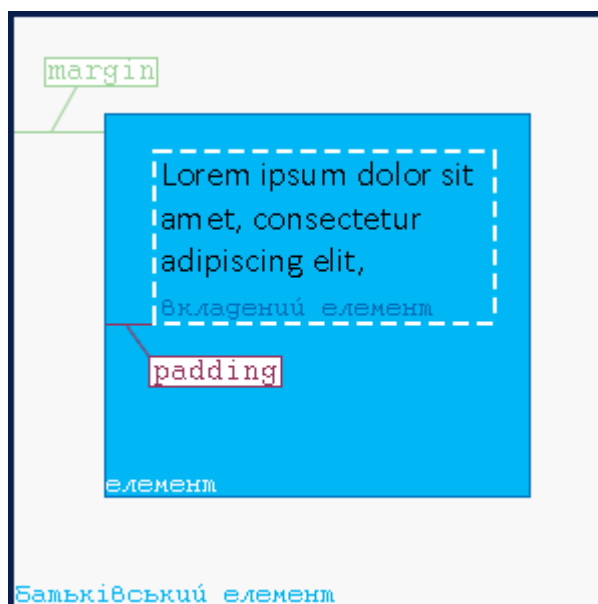


Рис 8.3 margin&padding

- **padding** - встановлює значення полів навколо вмісту елемента. Поле називається відстань від внутрішнього краю рамки елемента до вкладеного елемента

- Значення полів `margin` та `padding` можна вказувати в пікселях (px) та відсотках (%). Якщо значення полів вказані у відсотках, то вони обчислюються від значень полів батьківського елемента. Значення `auto` вказує, що розмір відступів буде автоматично розрахований браузером. `inherit` – значення властивості успадковується від батьківського елемента.

- **Наприклад:**

```
margin: 10px auto;
```

елемент буде розміщено з відступом 10 px згори та знизу від батьківського елемента та відцентровано по горизонталі;

- `border` - властивість `border` дозволяє одночасно встановити товщину, стиль і колір границі елемента. Для встановлення границі тільки з певних сторін елемента, використовуються властивості `border-top`, `border-bottom`, `border-left`, `border-right`. Стилі границь *рис 8.4*



Рис 8.4 Стилі границь

- `font` - універсальна властивість, яке дозволяє одночасно задати кілька характеристик шрифту і тексту.
- Наприклад оголошення `font: 25px Georgia;` задає елементу, що містить це оголошення розмір тексту 25px, шрифт Georgia.

- Також розмір тексту можна задати за допомогою властивості

`font-size`: абсолютний розмір | відносний розмір | значення | відсотки | `inherit`

- Для завдання абсолютного розміру використовуються наступні значення: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`. Відносний розмір шрифту задається значеннями `larger` і `smaller`. Також дозволяється використовувати будь-які допустимі одиниці CSS: `em` (висота шрифту елемента), `ex` (висота символу x), пункти (pt), пікселі (px), відсотки (%) і ін. За 100% береться розмір шрифту батьківського елемента. Негативні значення не допускаються.

- **font-weight** – задає насиченість шрифту. **bold** - напівжирний шрифт, **normal** - нормальне накреслення. Також допустимо використовувати умовні одиниці від 100 до 900. Значення **bolder** і **lighter** змінюють жирність відносно насиченості батька, відповідно, в більшу і меншу сторону.
- **text-align** - визначає горизонтальне вирівнювання тексту в межах елемента.
 - Можливі значення: **center** (вирівнювання по-центру), **justify** (вирівнювання по ширині), **left** (вирівнювання по лівому краю), **right** (вирівнювання по правому краю), **inherit** (унаслідкується значення батьківського елемента)
- **color** – встановлює колір тексту елемента. Значення: колір (ім'я, hex, rgb, hsl) або **inherit**.
- **background** - Універсальна властивість, дозволяє встановити одночасно до п'яти характеристик фону або **inherit**.
 - **background-attachment** (встановлює, чи буде прокручуватися фонове зображення разом з вмістом елемента)
 - **background-color** (визначає колір фону елемента)
 - **background-image** (встановлює фонове зображення для елемента)
 - **background-position** (задає початкове положення фонового зображення, встановленого за допомогою властивості **background-image**)
 - **background-repeat** (визначає, як буде повторюватися фонове зображення, встановлене за допомогою властивості **background-image**.)

Значення можуть йти в будь-якому порядку, браузер сам визначить, яке з них відповідає потрібній властивості.

Приклад:

```
background: url(images/картинка.png)
repeat-y #00f;
```

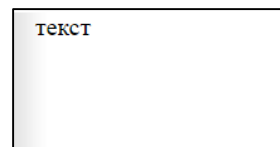


Рис 8.5

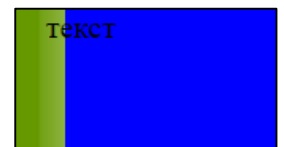


Рис 8.6

На *рис 8.6* можна побачити, що ширина картинки менша за розмір елемента, тому видно колір фону

- **height/ width** - встановлює висоту/ширину елементів.
 - Висота/ширина не включає товщину границь навколо елемента, значення відступів і полів.
 - Якщо вміст блоку перевищує зазначену висоту/ширину, то висота/ширина елемента залишиться незмінною, а вміст буде відображатися поверх нього.
 - Як значення приймаються будь-які одиниці довжини, прийняті в CSS - наприклад, пікселі (px), дюйми (in), пункти (pt) та ін.
 - Якщо значення полів вказані у відстоках, то вони обчислюються від значень полів батьківського елемента. Якщо батько явно не вказано, то в його якості виступає вікно браузера. **auto** встановлює висоту виходячи з вмісту елемента.

- border-collapse - встановлює, як відображати границі навколо комірок таблиці. За замовчуванням separate (рис 8.7), якщо для комірок встановлена рамка, то в місці стику комірок виходить лінія подвійної товщини. Значення collapse (рис 8.8) змушує браузер аналізувати подібні місця в таблиці і прибирати в ній подвійні лінії. Застосовується лише для тегу <table> або до елементів, у яких значення display встановлено як table або inline-table.

| Column 1 | Column 2 |
|----------|----------|
| AA | BA |
| AB | BB |

Рис 8.7 border-collapse:separate;

| Column 1 | Column 2 |
|----------|----------|
| AA | BA |
| AB | BB |

Рис 8.8 border-collapse:collapse;

- table-layout - визначає, як браузер повинен обчислювати ширину комірок таблиці, ґрунтуючись на їх вмісті. При значенні auto браузер завантажує всю таблицю, аналізує її для визначення розмірів осередків і тільки після цього відображає. fixed - ширина колонок в цьому випадку визначається або за допомогою тега <col>, або обчислюється на основі першого рядка. Якщо дані про форматування першого рядка таблиці з яких-небудь причин отримати неможливо, в цьому випадку таблиця ділиться на колонки рівної ширини.
- ❖ Псевдоклас hover визначає стиль елементу при наведенні на нього курсора миші, але при цьому елемент ще не активовано.

```
tr:hover
{
background: #800000;
color: #ffffff;
}
```

Рис 8.9

- ❖ Псевдоклас nth-child використовується для додавання стилю до елементів на основі нумерації в дереві елементів. (nth-child(2n)-всі парні елементи)

```
tr:nth-child(2n)
{
background: #DEEE5;
}
```

Рис 8.10

| Факультет кібернетики | | | |
|----------------------------------|--------------------------|------------|------------|
| Курс | 1 | | |
| Спеціальність | Інформатика | | |
| Кафедра | | | |
| Група | К-14 | | |
| Студент | Воробйова Аліна Ігорівна | | |
| ID | KNUCBoo1 | | |
| Предмет | Тип контролю | Оцінка/100 | Оцінка |
| Вступ до університетських студій | Залік | 60 | зараховано |
| Математичний аналіз | Екзамен | 90 | відмінно |
| Програмування | Екзамен | 76 | добре |

| Факультет кібернетики | | | |
|----------------------------------|--------------------------|------------|------------|
| Курс | 1 | | |
| Спеціальність | Інформатика | | |
| Кафедра | | | |
| Група | К-14 | | |
| Студент | Воробйова Аліна Ігорівна | | |
| ID | KNUCBoo1 | | |
| Предмет | Тип контролю | Оцінка/100 | Оцінка |
| Вступ до університетських студій | Залік | 60 | зараховано |
| Математичний аналіз | Екзамен | 90 | відмінно |
| Програмування | Екзамен | 76 | добре |

Рис 8.11 Вигляд результату трансформації у вікні браузера

9. Висновки

Отже, в умовах даної лабораторної роботи було створено програму, головною метою якої є аналіз вмісту xml-документу та трансформація результату у html-файл.

Вхідний xml-документ було створено власноруч у текстовому редакторі.

Аналіз вмісту документа представляє собою пошук інформації за критеріями, що обираються користувачем. Він реалізований трьома способами:

1. [LINQ](#)
2. [SAX](#)
3. [DOM](#)

Користувач власноруч обирає, який спосіб застосовувати.

Для реалізації аналізу також використовується антишиблон магічна кнопка та шаблон “Стратегія”.

Результат аналізу зберігається у новоствореному xml-файлі та за бажанням користувача відбувається його трансформція у html на основі xsl-файлу та за використанням таблиць стилів css. За замовчуванням результат трансформації відображається у вікні браузера.

Таким чином, в ході виконня цієї лабораторної роботи, завдання, які були поставлені - виконанні.

10. Джерела

- 1) <http://citforum.ck.ua/internet/xslt>
- 2) <https://msdn.microsoft.com>
- 3) <https://xsltdev.ru/>
- 4) <http://htmlbook.ru/>
- 5) <http://metanit.com/sharp>
- 6) <http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/iipXML/usax.html>