

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій

Кафедра програмних систем і технологій

ЗВІТ

з лабораторної роботи № 3

Тема: “Автоматизована система підземного паркування”

Дисципліна «Об’єктно-орієнтоване конструювання програм»

Підготував:

студент гр. ПІЗ-23(1)

Польнюк Руслан Дмитрович

Дата : _____

Перевірила:

доц. Зубик Л.В.

Тема: Патерни.

Мета роботи: ознайомитися з паттернами проектування.

Умова

Створити MVC ASP-застосунок. Для теми проекту з попереднього семестру (або іншої самостійно обраної теми, яка буде закріплена за студентом до кінця поточного семестру після її погодження з викладачами) розробити його модель, подання і контролер(-и).

Тема проекту: “Автоматизована система підземного паркування”.

Хід роботи:

Завдання 1:

В основі архітектурного шаблону MVC, лежать Model, View, Controller. Для початку реалізації цього шаблону потрібно створити моделі даних, які будуть взаємодіяти з контролерами, та зберігатися в базі даних.

- модель Place

```
Ссылка: 0
public class Place
{
    Ссылка: 0
    public int Id { get; set; }
    Ссылка: 0
    public int Number { get; set; }
}
```

- модель Order

```
Ссылка: 0
public class Order
{
    Ссылка: 0
    public int Id { get; set; }
    Ссылка: 0
    public int Hourse { get; set; }
    Ссылка: 0
    public int PlaceId { get; set; }
}
```

Після створення моделей, потрібно написати взаємодію застосунку з базою даних, в моєму застосунку за це буде відповідати orm Entity Framework Core. Для цього потрібно створити клас під назвою ParkingDbContext, який

буде містити всі базові налаштування для взаємодії з базою даних.

```
Ссылка: 2
public class ParkingDbContext : DbContext
{
    Ссылка: 0
    public DbSet<Place> Places { get; set; }
    Ссылка: 0
    public DbSet<Order> Orders { get; set; }

    Ссылка: 0
    public ParkingDbContext(DbContextOptions<ParkingDbContext> options) : base(options)
    {
    }

    Ссылка: 0
    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);
    }
}
```

Після цього можна створити імплементацію паттерну “Strategy”, для цього потрібно створити інтерфейс, який буде мати методи з певною сигнатурою, та сервіс який буде реалізовувати даний інтерфейс.

```
Ссылка: 0
public interface IParkingRepository
{
    Ссылка: 0
    IEnumerable<Place> GetavailablePlaces();
    Ссылка: 0
    void MakeReservation(Order order);
}
```

```

Ссылка: 1
public class ParkingService : IParkingRepository
{
    private readonly ParkingDbContext _context;

    Ссылка: 0
    public ParkingService(ParkingDbContext context)
    {
        _context = context;
    }

    Ссылка: 1
    public IEnumerable<Place> GetavailablePlaces()
    {
        return _context.Places.ToList();
    }

    Ссылка: 1
    public void MakeReservation(Order order)
    {
        var makeOrder = new Order
        {
            Id = order.Id,
            Hourse = order.Hourse,
            PlaceId = order.PlaceId
        };

        _context.Orders.Add(makeOrder);
        _context.SaveChanges();
    }
}

```

Після імплементації інтерфейсу, потрібно створити контролери, які будуть комунікувати з моделями через сервіси, та відправляти потрібні дані на відображення.

```

Ссылка: 0
public class BaseController : Controller
{
    Ссылка: 0
    public IActionResult Index()
    {
        return View();
    }
}

```

Ссылка: 1

```
public class OrderController : Controller
{
    private readonly IParkingRepository _repository;

    Ссылка: 0
    public OrderController(IParkingRepository repository)
    {
        _repository = repository;
    }

    Ссылка: 0
    public IActionResult Index()
    {
        return View();
    }

    [HttpGet]
    Ссылка: 0
    public IActionResult GetavailablePlaces()
    {
        var result = _repository.GetavailablePlaces();
        return View(result);
    }
}
```

[HttpGet]

Ссылка: 0

```
public IActionResult MakeReservation(int id)
{
    ViewBag.Id = id;
    return View();
}
```

[HttpPost]

Ссылка: 0

```
public string MakeReservation(Order order)
{
    _repository.MakeReservation(order);
    return "Reservasition is done";
}
```

Останній крок який потрібно зробити це написати відображення.

Відповідно код для відображення контролерів на прикладі Order:

```

1  <body>
2  <div>
3      <table>
4          <tr>
5              <td>GetAllOrders</td>
6              <td>AddOrder</td>
7          </tr>
8          <tr>
9              <td>
10                 <form action="@Url.Action("GetAllOrders", "FinalOrder")" method="get">
11                     <input type="hidden"/>
12                     <button type="submit">GetAllOrders</button>
13                 </form>
14             </td>
15             <td>
16                 <form action="@Url.Action("AddOrder", "FinalOrder")" method="get">
17                     <input type="hidden" name="Id"/>
18                     <button type="submit">AddOrder</button>
19                 </form>
20             </td>
21          </tr>
22      </table>
23  </div>
24  </body>

```

```

1  <body>
2  <div>
3      <table>
4          <tr>
5              <td>Id</td>
6              <td>Number</td>
7          </tr>
8          @foreach(var value in Model)
9          {
10             <tr>
11                 <td>@value.Id</td>
12                 <td>@value.Number</td>
13             </tr>
14         }
15     </table>
16 </div>
17 </body>

```

```

1  <form method="post" action="">
2      <input type="hidden" value="@ViewBag.Id" name="Id"/>
3      <table>
4          <tr>
5              <td><p>Enter Id:</p></td>
6              <td><input type="text" name="Id"/></td>
7          </tr>
8          <tr>
9              <td><p>Enter Number:</p></td>
10             <td><input type="text" name="Number"/></td>
11          </tr>
12          <tr>
13              <td><input type="submit"/>MakeReservation</td>
14          </tr>
15      </table>
16 </form>

```

Скріншот результатів:

lab2 Home Order

lab2 Home Order

GetavailablePlaces MakeReservation

GetavailablePlaces MakeReservation

lab2 Home Order

Id Number

1 5

lab2 Home Order

Enter Id:

Enter Number:

MakeReservation

Reservasion is done

Завдання 2:

Для реалізації завдання № 2 було обрано вже знайомий паттерн “Strategy”. Реалізація буде за допомоги winforms. Завданням є розрахунок периметру правильних багатокутників.

Для початку потрібно створити клас багатокутник, який буде використовуватися в реалізації.

```
Ссылка: 1
public class Polygon
{
    Ссылка: 1
    public decimal NumberOfSide { get; set; }
    Ссылка: 1
    public decimal Lenght { get; set; }

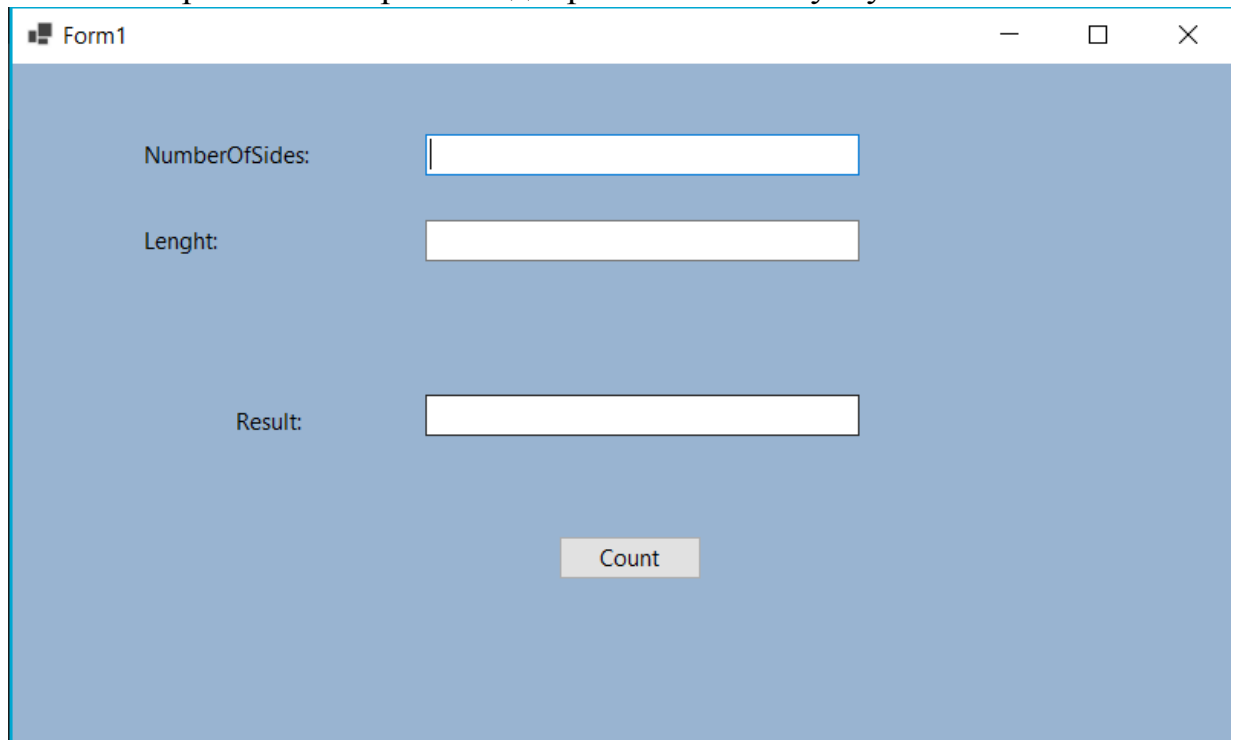
    Ссылка: 0
    public Polygon(decimal numberOfSide, decimal lenght)
    {
        NumberOfSide = numberOfSide;
        Lenght = lenght;
    }
}
```

Наступним кроком є створення інтерфейсу, який буде реалізовуватися сервісом, та сам сервіс.

```
Ссылка: 0
public interface IStrategy
{
    Ссылка: 0
    decimal Perimetr(Polygon polygon);
}
```

```
Ссылка: 0
public class StrategyService : IStrategy
{
    Ссылка: 1
    public decimal Perimetr(Polygon polygon)
    {
        decimal result = polygon.NumberOfSide * polygon.Lenght;
        return result;
    }
}
```

Останнім кроком є створення відображення застосунку.



The screenshot shows a Windows application window titled "Form1". The window has a light blue background and contains three input fields with labels "NumberOfSides:", "Lenght:", and "Result:". Below the input fields is a button labeled "Count".


```

StrategyService service = new StrategyService();
Ссылка: 1
public Form1()
{
    InitializeComponent();
}

Ссылка: 1
private void button1_Click(object sender, EventArgs e)
{
    decimal numberOfSides = Convert.ToDecimal(textBox1.Text);
    decimal lenght = Convert.ToDecimal(textBox2.Text);
    Polygon polygon = new Polygon(numberOfSides, lenght);

    decimal result = service.Perimetr(polygon);

    textBox3.Text = result.ToString();
}

```

Скріншот результатів:

The screenshot shows a Windows Forms application window titled "Form1". The window has a light blue background. It contains three text boxes: "NumberOfSides" with the value "5", "Lenght" with the value "5", and "Result" with the value "25". Below these text boxes is a button labeled "Count".

Висновки:

Під час виконання цієї лабораторної роботи я ознайомився з патернами проектування, які входять в Gof, та допомагають створювати більш гнучкі та великі системи. Для першого завдання я розробив веб-застосунок за допомогою Asp net Core Mvc згідно своєї теми. Застосував ORM Ef core для роботи з БД, та паттерн Strategy. Для 2 завдання я розробив десктопний застосунок за допомогою Winforms