Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

З лабораторної роботи №3

з курсу

«Основи розробки програмного забезпечення на платформі Microsoft.NET»

Виконав студент *ІП-01 Галько Міла*
(шифр, ПІБ)

Перевірила *Ліщук К.І.*
(ПІБ)

Київ 2022

Комп‘ютерний практикум № 3. Шаблони проектування. Породжуючі шаблони

Мета:

- ознайомитися з основними шаблонами проектування, навчитися застосовувати їх при проектуванні і розробці ПЗ.

Постановка задачі комп‘ютерного практикуму № 3

При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:

1) Вивчити породжуючи патерни. Знати загальну характеристику та призначення кожного з них, особливості реалізації кожного з породжуючих патернів та випадки їх застосування.
2) Реалізувати задачу згідно варіанту №1. Розробити інтерфейси та класи з застосування одного або декількох патернів. Повністю реалізувати методи, пов‘язані з реалізацією обраного патерну.
3) Повністю описати архітектуру проекту (призначення методів та класів), особливості реалізації обраного патерну. Для кожного патерну необхідно вказати основні класи та їх призначення,
4) Навести UML-діаграму класів

Варіант №1

Реалізувати задачу «Електронний документообіг». Оброблюються документи декількох типів (наприклад, лист, службова записка, розпорядження, наказ, заявка на ресурс тощо). Кожен документ містить номер, дату та інформацію про документ. Крім того, в листи можуть бути як вхідними, так і вихідними та містять кореспондента, від кого надійшов лист або надсилається. Накази містять підрозділ, строк виконання та відповідального виконавця. Розпорядження – тільки підрозділ та строк виконання. Заявки на ресурси містять співробітника, котрому необхідно забезпечити доступ до ресурсів, перелік ресурсів.

Програмний код:

Посилання на github: *https://github.com/MilaHalko/C4_.NET/tree/Lab3*
Також код доданий в кінці документу.

Опис проблематики та архітектури проекту:

Оскільки стикаємося з проблемою, що заздалегідь невідомий кінцевий продукт; та за формулюванням задачі можуть бути і інші варіанти документів, що можуть бути додані до системи пізніше, то загалом зрозуміло, що код повинен бути більш універсальним для подальшого додання нових класів.

Отже, фабричний метод стане найкращим способом реалізації. Даний патерн надає змогу зробити код універсальним, завдяки тому, що немає прив’язки до конкретних класів. Для реалізації потребуємо загальний абстрактний клас – DocCreator. Від ного будуть унаслідуватись усі інші Cretor-и для створення кожного з файлів (спадкоємці Doc).

Єдина проблема – файли мають різні атрибути. В даній роботі для передачі усіх необхідних значень для створення файлів був створений абстрактний клас DocArgs та його спадкоємці (роль сховища). Об’єкт класу DocArgs буде переданий у конструктор спадкоємця DocCreator-а для подальшого вилучення значень. В залежності від переданого спадкоємця DocArgs буде створений певний спадкоємець DocCreator.

Doc – абстрактний клас узагальнюючий поняття «Документ». Має загальні атрибути: id, date, info.

Memo, Letter, Order, Decree, ResourceRequest – спадкоємці класу Doc, кожен є представником певного виду документу із своїми відповідними атрибутами.

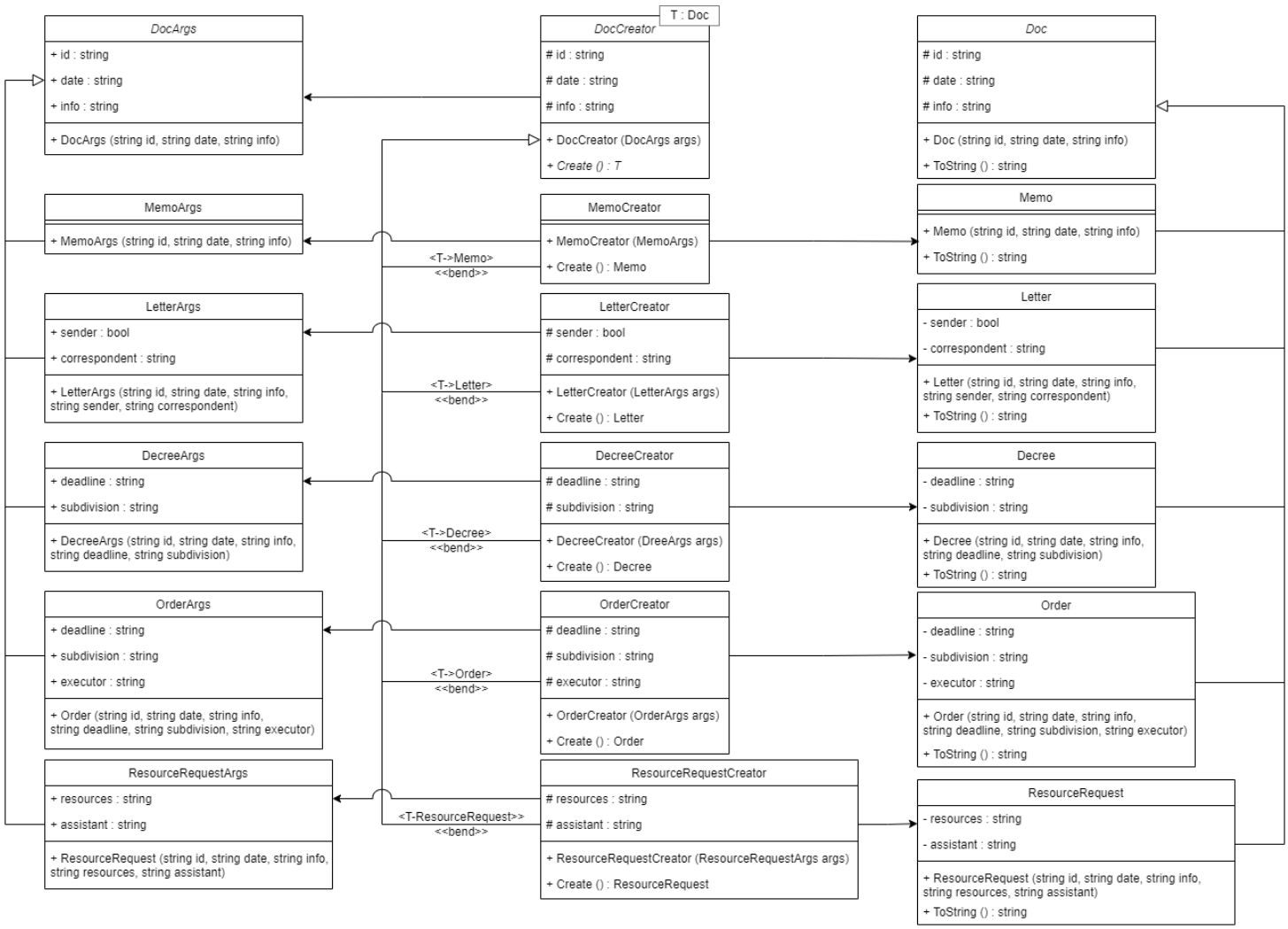
DocCreator – абстаракний клас, відповідає за повернення новостворених файлів через абстрактний метод Create().

MemoCreator, LetterCreator, OrderCreator, DecreeCreator, ResourceRequestCreator – конкретні створювачі; спадкоємці DocCreator; реалізують метод Create(); мають атрибути відповідно до файлу для зберігання значень після їх безпосереднього створення.

T Create () – абстрактний метод класу DocCreator, що створює та повертає T – один із спадкоємців класу Doc ().

DocArgs – абстрактний клас, відіграє роль сховища значень для створення файлу.

MemoArgs, LetterArgs, OrderArgs, DecreeArgs, ResourceRequestArgs – спадкоємці DocArgs; мають атрибути – параметри для майбутнього створення файлу; є необхідним параметром при створенні спадкоємця DocCreator, оскільки значення його атрибутів далі будуть зберігатися у даному класі.



Маємо зв'язки типу:

Generalization – звичайне наслідування класів;

Association – класи потребують об'єкти інших класів у методах лише для зібрання значень, але сама зміна переданих об'єктів не впливає на подальшу зміну залежного об'єкту.

Program.cs та скрін виконання програми

```
using Lab3.Creators;
using Lab3.DocsArgs;
using Lab3.Docs;

namespace Lab3
{

    class Program
    {
        static void Main(string[] args)
        {
            List<Docs.Doc> docs = new();

            var mArgs = new MemoArgs("rst-3Re", "12/12/2022", "Memo to stuff.");
            var mCreator = new MemoCreator(mArgs);
            Doc m = mCreator.Create();
            docs.Add(m);

            var lArgs = new LetterArgs("hd3_5", "18/04/2022", "Letter to officer, IMPOrTAnt!", fals
        e, "Mr.James");
            var lCreator = new LetterCreator(lArgs);
            Doc l = lCreator.Create();
            docs.Add(l);

            var dArgs = new DecreeArgs("4gT-
        13e", "09/09/2021", "Stop rewriting memo!", "12/12/2022", "SD-13");
            var dCreator = new DecreeCreator(dArgs);
            Doc d = dCreator.Create();
            docs.Add(d);

            var oArgs = new OrderArgs("d1R4", "01/01/2018", "Order13/4-1", "15/06/2020", "SD-
        01A", "Lia Charms");
            var oCreator = new OrderCreator(oArgs);
            Doc o = oCreator.Create();
            docs.Add(o);

            var rrArgs = new ResourceRequestArgs("5oR_7", "18/10/2019", "Institute resources", "pen
        s, tables, books, tablets", "Louis Baroette");
            var rrCreator = new ResourceRequestCreator(rrArgs);
            Doc rr = rrCreator.Create();
            docs.Add(rr);

            foreach (Docs.Doc doc in docs)
            {
                Console.WriteLine(doc);
            }
        }
    }
}
```

```
Microsoft Visual Studio Debug Console
Doc.Memo #rst-3Re by 12/12/2022
Information: Memo to stuff.

Doc.Letter #hd3_5 by 18/04/2022
Receiver: Mr.James
Information: Letter to officer, IMPOrTAnt!

Doc.Decree #4gT-13e by 09/09/2021
Information: Stop rewriting memo!
Deadline: 12/12/2022
Subdivision: SD-13

Doc.Order #d1R4 by 01/01/2018
Information: Order13/4-1
Deadline: 15/06/2020
Subdivision: SD-01A
Executor: Lia Charms

Doc.ResourseRequest #5oR_7 by 18/10/2019
Information: Institute resources
Assistant: Louis Baroette
Resources: pens, tables, books, tablets
```

Doc.cs

```
namespace Lab3.Docs
{
    public abstract class Doc
    {
        protected string id;
        protected string date;
        protected string info;
        public Doc(string id, string date, string info)
        {
            this.id = id;
            this.date = date;
            this.info = info;
        }
        public override string ToString()
        {
            return $"Document #{id} by {date}\n" +
                   $"Information: {info}\n";
        }
    }

    class Memo : Doc
    {
        public Memo(string id, string date, string info) : base(id, date, info) { }
        public override string ToString()
        {
            return $"Doc.Memo #{id} by {date}\n" +
                   $"Information: {info}\n";
        }
    }

    class Letter : Doc
    {
        private bool sender;
        private string correspondent;
        public Letter(string id, string date, string info, bool sender, string correspondent)
            : base(id, date, info)
        {
            this.sender = sender;
            this.correspondent = correspondent;
        }
        public override string ToString()
        {
            string status = sender ? "Sender" : "Receiver";
            return $"Doc.Letter #{id} by {date}\n" +
                   $"{status}: {correspondent}\n" +
                   $"Information: {info}\n";
        }
    }

    class Decree : Doc
    {
        private string deadline;
        private string subdivision;
        public Decree(string id, string date, string info, string deadline,
            string subdivision) : base(id, date, info)
        {
            this.deadline = deadline;
            this.subdivision = subdivision;
        }
        public override string ToString()
        {
            return $"Doc.Decree #{id} by {date}\n" +
                   $"Information: {info}\n" +
                   $"Deadline: {deadline}\n" +
                   $"Subdivision: {subdivision}\n";
        }
    }
```

```
    class Order : Doc
    {
        private string subdivision;
        private string deadline;
        private string executor;
        public Order(string id, string date, string info, string deadline,
            string subdivision, string executor)
            : base(id, date, info)
        {
            this.executor = executor;
            this.deadline = deadline;
            this.subdivision= subdivision;
        }
        public override string ToString()
        {
            return $"Doc.Order #{id} by {date}\n" +
                   $"Information: {info}\n" +
                   $"Deadline: {deadline}\n" +
                   $"Subdivision: {subdivision}\n" +
                   $"Executor: {executor}\n";
        }
    }

    class ResourceRequest : Doc
    {
        private string resources;
        private string assistant;
        public ResourceRequest(string id, string date, string info,
            string resouces, string assistant) : base(id, date, info)
        {
            this.resources = resouces;
            this.assistant = assistant;
        }
        public override string ToString()
        {
            return $"Doc.ResourseRequest #{id} by {date}\n" +
                   $"Information: {info}\n" +
                   $"Assistant: {assistant}\n" +
                   $"Resources: {resources}\n";
        }
    }
}
```

DocArgs.cs

```
namespace Lab3.DocsArgs
{
    abstract class DocArgs
    {
        public string id { get; set; }
        public string date { get; set; }
        public string info { get; set; }
        public DocArgs(string id, string date, string info)
        {
            this.id = id;
            this.date = date;
            this.info = info;
        }
    }
    class MemoArgs : DocArgs
    {
        public MemoArgs(string id, string date, string info)
            : base(id, date, info) { }
    }
    class LetterArgs : DocArgs
    {
        public bool sender { get; set; }
        public string correspondent { get; set; }
        public LetterArgs(string id, string date, string info,
            bool sender, string correspondent)
            : base(id, date, info)
        {
            this.sender = sender;
            this.correspondent = correspondent;
        }
    }
    class DecreeArgs : DocArgs
    {
        public string deadline { get; set; }
        public string subdivision { get; set; }
        public DecreeArgs(string id, string date, string info,
            string deadline, string subdivision) : base(id, date, info)
        {
            this.deadline = deadline;
            this.subdivision = subdivision;
        }
    }
    class OrderArgs : DocArgs
    {
        public string deadline { get; set; }
        public string subdivision { get; set; }
        public string executor { get; set; }
        public OrderArgs(string id, string date, string info,
            string deadline, string subdivision, string executor): base(id, date, info)
        {
            this.executor = executor;
            this.subdivision = subdivision;
            this.deadline = deadline;
        }
    }
    class ResourceRequestArgs : DocArgs
    {
        public string resources { get; set; }
        public string assistant { get; set; }
        public ResourceRequestArgs(string id, string date, string info,
            string resouces, string assistant) : base(id, date, info)
        {
            this.resources = resouces;
            this.assistant = assistant;
        }
    }
}
```

Creator.cs

```
using Lab3.DocsArgs;
using Lab3.Docs;

namespace Lab3.Creators
{
    abstract class DocCreator<T> where T : Docs.Doc
    {
        protected string id;
        protected string date;
        protected string info;
        public DocCreator(DocArgs args)
        {
            id = args.id;
            date = args.date;
            info = args.info;
        }
        abstract public T Create();
    }
    class MemoCreator : DocCreator<Memo>
    {
        public MemoCreator(MemoArgs args) : base(args)
        {
        }
        public override Memo Create()
        {
            return new Memo(id, date, info);
        }
    }

    class LetterCreator : DocCreator<Letter>
    {
        protected bool sender;
        protected string correspondent;
        public LetterCreator(LetterArgs args) : base(args)
        {
            sender = args.sender;
            correspondent = args.correspondent;
        }
        public override Letter Create()
        {
            return new Letter(id, date, info, sender, correspondent);
        }
    }

    class DecreeCreator : DocCreator<Decree>
    {
        protected string deadline;
        protected string subdivision;
        public DecreeCreator(DecreeArgs args) : base(args)
        {
            deadline = args.deadline;
            subdivision = args.subdivision;
        }

        public override Decree Create()
        {
            return new Decree(id, date, info, deadline, subdivision);
        }
    }

    class OrderCreator : DocCreator<Order>
    {
        protected string deadline;
        protected string subdivision;
        protected string executor;
        public OrderCreator(OrderArgs args) : base(args)
        {
```

```
            deadline = args.deadline;
            subdivision = args.subdivision;
            executor = args.executor;
        }

        public override Order Create()
        {
            return new Order(id, date, info, deadline, subdivision, executor);
        }
    }

    class ResourceRequestCreator : DocCreator<ResourceRequest>
    {
        protected string resources;
        protected string assistant;
        public ResourceRequestCreator(ResourceRequestArgs args) : base(args)
        {
            resources = args.resources;
            assistant = args.assistant;
        }
        public override ResourceRequest Create()
        {
            return new ResourceRequest(id, date, info, resources, assistant);
        }
    }

}
```