МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 4

з дисципліни “Основи програмування”
тема “Породжуючі шаблони проектування”

Виконав	Перевірив
студент II курсу	“____” “_____” 20___ р.
групи КП-01	викладач
Беліцький Олександр Сергійович (*прізвище, ім’я, по батькові*)	Заболотня Тетяна Миколаївна (*прізвище, ім’я, по батькові*)

варіант № 3

Київ 2021

Постановка завдання

Завдання 1.

За допомогою шаблону проєктування реалізувати процес випуску книжок за заготовленими раніше матрицями. Кожна книга характеризується іменем автора, видавництвом, де вона була надрукована, серією та номером видання.

Завдання 2.

За допомогою шаблону проєктування реалізувати процес формування штучної водойми, який складається з таких етапів як побудова котловану, заповнення його водою, висадження рослин, запуск водних мешканців. В залежності від типу ділянки котлован може бути як бетонним (каркасним), так і виконаний як пластикова чаша чи просто яма, застелена плівкою.

Аналіз вимог і проектування

Завдання 1.

У даній задачі доцільно використати шаблон Прототип.

Вибір впав саме на цей шаблон, адже він дозволяє створювати об'єкти на основі раніше створених об'єктів-прототипів. Тобто, по суті, цей патерн пропонує техніку клонування об'єктів.

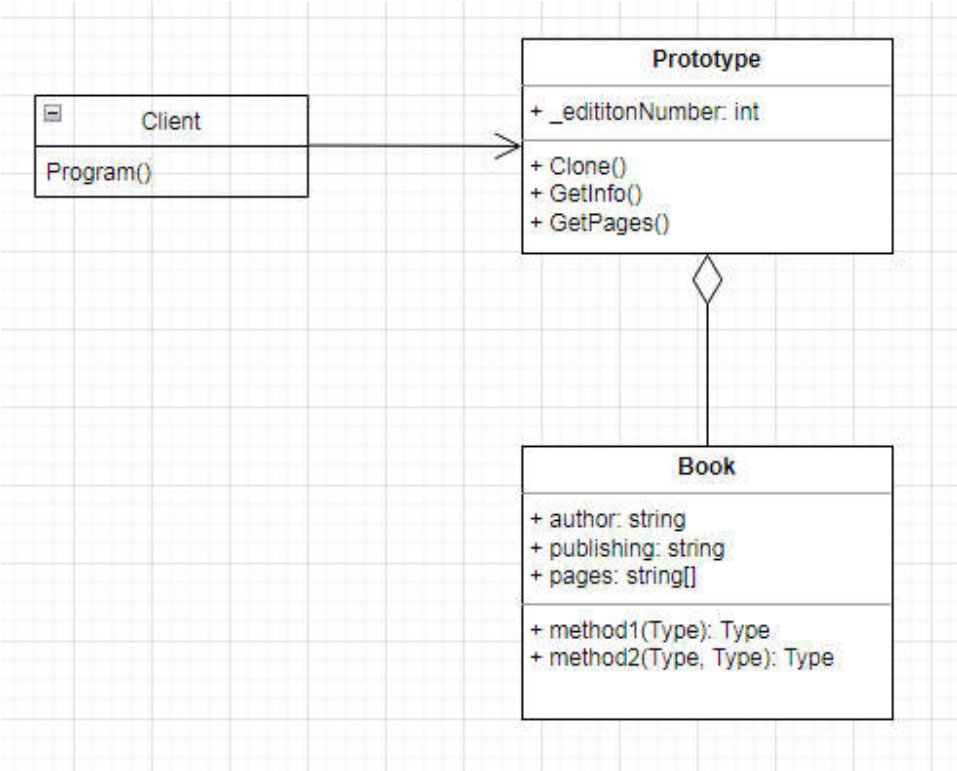


рис.1 UML-діаграма класів

Завдання 2.

У даній задачі доцільно використати шаблон Будувальник.

Паттерн Будувальник інкапсулює створення об'єкта та дозволяє розділити його на різні етапи.

В даній задачі нам потрібно розбити процес створення водойми на декілька етапів та забезпечити можливість варіативності водойм за видом покриття котловану.

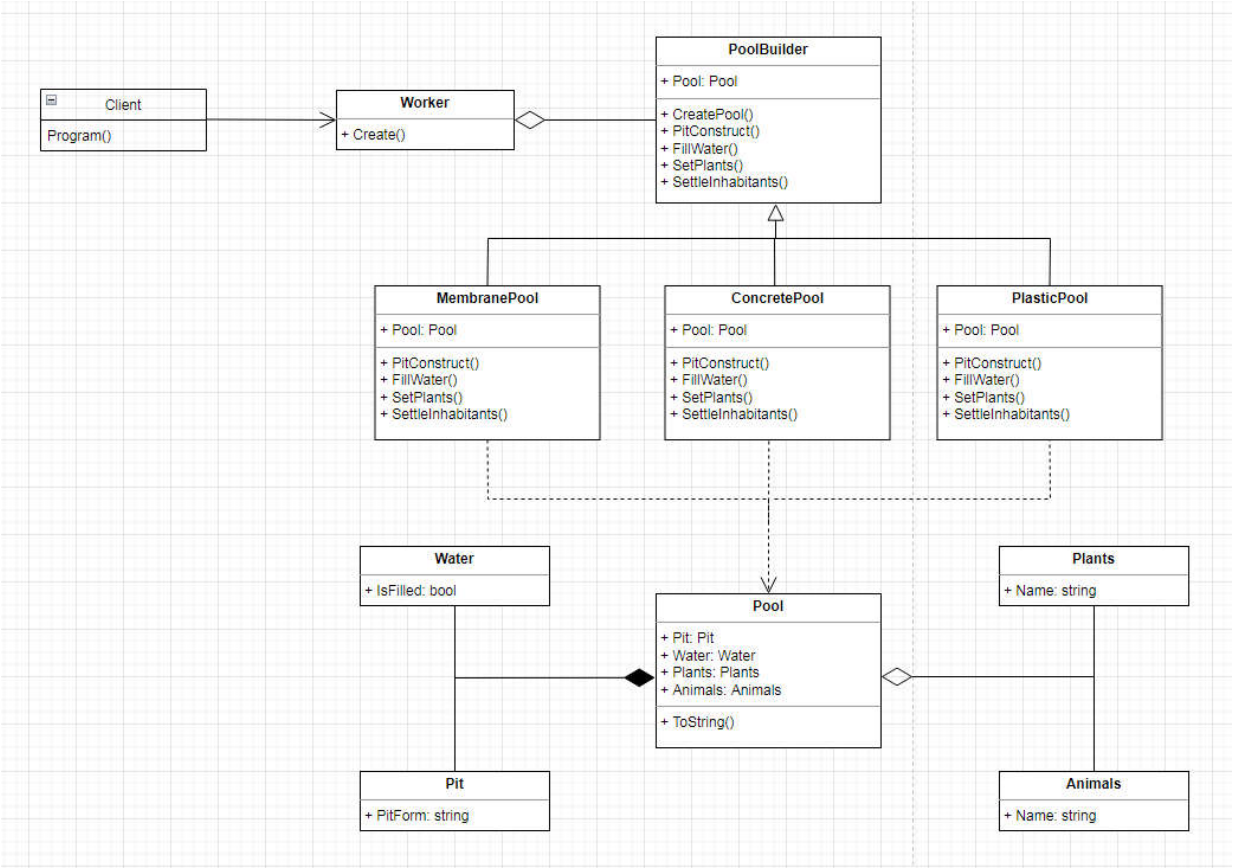


рис.2 UML-діаграма класів

Тексти коду програм

Завдання 1.

Program.cs

```
using System;

class Program
{
    static void Main(string[] args)
    {
        string[] pages = new string[] { "page1", "page2", "page3" };
        Prototype book1 = new Book(1, "Lev", "Kyiv", pages);
        Prototype clonedBook1 =  book1.Clone();
        book1.GetInfo();
        book1.GetPages();
        Console.WriteLine("------");
        clonedBook1.GetInfo();
        clonedBook1.GetPages();

    }
}

abstract class Prototype
{
    private int _editionNumber;
    public Prototype(int id)
    {
        this._editionNumber = id;
    }
    public int EditionNumber
    {
        get { return _editionNumber; }
    }
    public abstract Prototype Clone();
    public abstract void GetInfo();
    public abstract void GetPages();
}

class Book : Prototype
{
    private string _author;
    private string _publishing;
    string[] pages;

    public Book(int editionNumber, string author, string publishing, string[] pages) :
base(editionNumber)
    {
        this._author = author;
        this._publishing = publishing;
        this.pages = pages;
    }
    public override Prototype Clone()
    {
        return this.MemberwiseClone() as Prototype;
    }

    public override void GetInfo()
    {
        Console.WriteLine($"Author : {this._author}\nPublishing : {this._publishing}\nNumber of
publishing : {this.EditionNumber}\nNumber of pages : {this.pages.Length}");
    }

    public override void GetPages()
    {
        Console.WriteLine("Pages: ");
        foreach(var item in pages)
            Console.WriteLine($"\t{item}");
```

```
    }
}
```

Завдання 2.

Program.cs

```
using System;
using System.Text;

class Program
{
    static void Main(string[] args)
    {
        Worker worker = new Worker();
        PoolBuilder builder = new ConcretePool();
        Pool concretePool = worker.Create(builder);
        Console.WriteLine(concretePool.ToString());

        builder = new PlasticPool();
        Pool plasticPool = worker.Create(builder);
        Console.WriteLine(plasticPool.ToString());

        builder = new MembranePool();
        Pool membranePool = worker.Create(builder);
        Console.WriteLine(membranePool.ToString());
    }
}

abstract class PoolBuilder
{
    public Pool pool { get; private set; }
    public void CreatePool()
    {
        pool = new Pool();
    }
    public abstract void PitConstruction();
    public abstract void FillWater();
    public abstract void SetPlants();
    public abstract void SettleInhabitants();
}

class Worker
{
    public Pool Create(PoolBuilder poolBuilder)
    {
        poolBuilder.CreatePool();
        poolBuilder.PitConstruction();
        poolBuilder.FillWater();
        poolBuilder.SetPlants();
        poolBuilder.SettleInhabitants();
        return poolBuilder.pool;
    }
}

class Pit
{
    public string pitForm { get; set; }
}
class Water
{
    public bool IsFilled { get; set; }
}
class Plants
{
    public string Name { get; set; }
}
```

```
class Animal
{
    public string Name { get; set; }
}

class Pool
{
    public Pit pit { get; set; }
    public Water water { get; set; }
    public Plants plants { get; set; }
    public Animal animals { get; set; }

    public override string ToString()
    {
        StringBuilder sb = new StringBuilder();

        if (pit != null)
            sb.Append("Pit: " + pit.pitForm + "\n");
        if (water != null)
            sb.Append("Is filled:" + water.IsFilled + "\n");
        if (plants != null)
            sb.Append("plants: " + plants.Name + " \n");
        if (animals != null)
            sb.Append("animals: " + animals.Name + " \n");
        return sb.ToString();
    }
}

class ConcretePool : PoolBuilder
{
    public override void PitConstruction()
    {
        this.pool.pit = new Pit { pitForm = "Concrete" };
    }

    public override void FillWater()
    {
        this.pool.water = new Water { IsFilled = true };
    }

    public override void SetPlants()
    {
        this.pool.plants = new Plants { Name = "Lilies" };
    }

    public override void SettleInhabitants()
    {
        this.pool.animals = new Animal { Name = "Frogs" };
    }
}

class PlasticPool : PoolBuilder
{
    public override void PitConstruction()
    {
        this.pool.pit = new Pit { pitForm = "Plastic" };
    }

    public override void FillWater()
    {
        this.pool.water = new Water { IsFilled = true };
    }

    public override void SetPlants()
    {
        this.pool.plants = new Plants { Name = "Sea cucumbers" };
    }

    public override void SettleInhabitants()
    {
        this.pool.animals = new Animal { Name = "Axolotls" };
```

```
    }
}

class MembranePool : PoolBuilder
{
    public override void PitConstruction()
    {
        this.pool.pit = new Pit { pitForm = "Membrane" };
    }

    public override void FillWater()
    {
        this.pool.water = new Water { IsFilled = true };
    }

    public override void SetPlants()
    {
        this.pool.plants = new Plants { Name = "Reeds" };
    }

    public override void SettleInhabitants()
    {
        this.pool.animals = new Animal { Name = "Fishes" };
    }
}
```

Приклади результатів

Завдання 1.

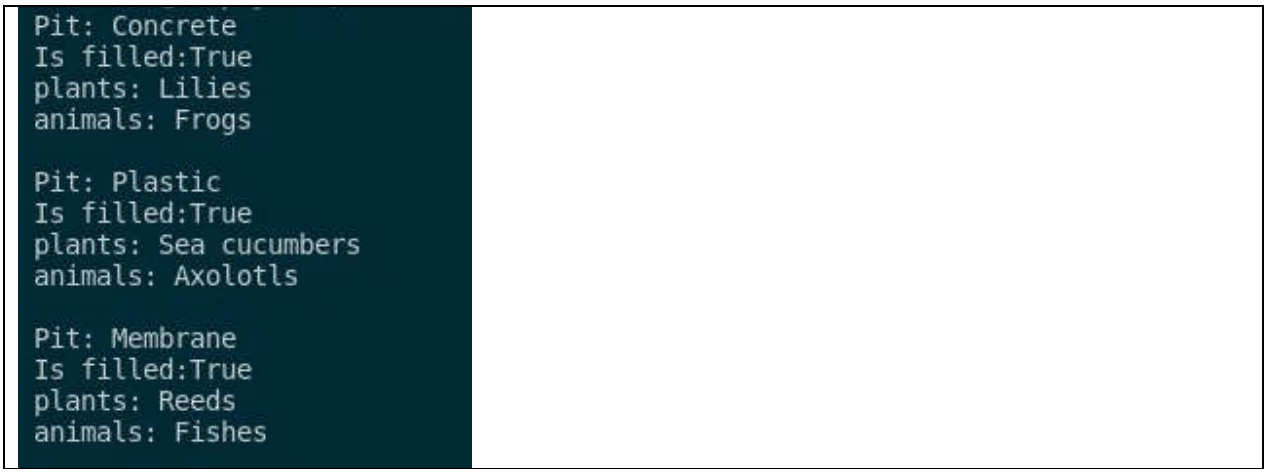
Програма дозволяє клонувати книжки:

```
        Prototype book1 = new Book(1,
"Lev", "Kyiv", pages);
        Prototype clonedBook1 =
book1.Clone();
        book1.GetInfo();
        book1.GetPages();
        Console.WriteLine("------");
        clonedBook1.GetInfo();
        clonedBook1.GetPages();
```

Цей механізм якраз і застосовується у книгодрукуванні. Коли друкар передає на станок друкувати книгу, то він не відправляє кожен екземпляр окремо, а вказує кількість копій, а принтер вже відпрацьовує потрібну кількість раз.

Завдання 2.

Створимо всі три види басейну. Приклад роботи програми:



У нас є працівник, що забезпечує роботу шаблону будувальника, нибі це покрокова інструкція:

```
class Worker
{
    public Pool Create(PoolBuilder poolBuilder)
    {
        poolBuilder.CreatePool();
        poolBuilder.PitConstruction();
        poolBuilder.FillWater();
        poolBuilder.SetPlants();
        poolBuilder.SettleInhabitants();
        return poolBuilder.pool;
    }
}
```

В залежності від типу ділянки програмою передбачено 3 класи:

```
class ConcretePool : PoolBuilder
{
    public override void
PitConstruction()
    {
        this.pool.pit = new Pit {
pitForm = "Concrete" };
    }

    public override void
FillWater()
    {
        this.pool.water = new
Water { IsFilled = true };
    }

    public override void
SetPlants()
    {
        this.pool.plants = new
Plants { Name = "Lilies" };
    }

    public override void
SettleInhabitants()
    {
        this.pool.animals = new
```

```
class PlasticPool : PoolBuilder
{
    public override void
PitConstruction()
    {
        this.pool.pit = new Pit {
pitForm = "Plastic" };
    }

    public override void
FillWater()
    {
        this.pool.water = new
Water { IsFilled = true };
    }

    public override void
SetPlants()
    {
        this.pool.plants = new
Plants { Name = "Sea cucumbers" };
    }

    public override void
SettleInhabitants()
    {
        this.pool.animals = new
```

```
class MembranePool : PoolBuilder
{
    public override void
PitConstruction()
    {
        this.pool.pit = new Pit {
pitForm = "Membrane" };
    }

    public override void
FillWater()
    {
        this.pool.water = new
Water { IsFilled = true };
    }

    public override void
SetPlants()
    {
        this.pool.plants = new
Plants { Name = "Reeds" };
    }

    public override void
SettleInhabitants()
    {
        this.pool.animals = new
```

Animal { Name = "Frogs" };
    }
}
	Animal { Name = "Axolotls" };
	    }
	}
		Animal { Name = "Fishes" };
		    }
		}

Користувач обирає потрібний йому тип, і програма за допомогою робітника самостійно виконує побудову басейну.

Висновки

Виконавши дану лабораторну роботу було проведено вивчення породжуючих шаблонів, їх особливостей, основних задач, принципів роботи та варіанти для вдалого використання.

Я визначив для себе, що породжуючі шаблони - це патерни, які мають справу з механізмами створення об'єкта та намагаються створити об'єкти в порядку, що підходить до ситуації.

Окремо детально було розглянуто патерни Прототип та Будувальник. Визначив, що їх можна охарактеризувати як:

- Прототип, по суті, пропонує техніку клонування об'єктів.
- Будувальник - інструкція(покроковий посібник) по створенню необхідного об'єкта.

Компіляція всього коду відбувалася за допомогою утиліти dotnet.