v

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №**1

з дисципліни “Основи об’єктно-орієнтованого програмування”

тема “ Реалізація структурних шаблонів проектування ”

|  |  |  |
| --- | --- | --- |
| Виконала  студентка II курсу  групи КП-83  Дзьоба Уляна Олегівна  (*прізвище, ім’я, по батькові*)  варіант №5 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2020 р.  викладач  Юсин Яків Олексійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Ознайомлення з основними характеристиками шаблонів «Декоратор», «Компонувальник» та «Міст», запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду

**Постановка завдання**

1. На виставці собак всі учасники мають такі спільні характеристики як зріст, вага, вік, порода, ім’я хазяїна. Однак, деякі породи собак можуть мати додаткові обов’язкові характеристики: наприклад, куповані хвіст та вушка, додаткові щеплення тощо. Крім того, різняться також і функції, природні для собак: одні є мисливськими, інші – бійцівськими. Отже, крім загальних команд, вони мають вміти виконувати і ті, що притаманні лише їх породі. Створити базовий клас «Собака» та декілька класів, що характеризують певні породи. Продемонструвати програмну реалізацію даних класів.
2. За допомогою шаблона проектування реалізувати механізм видання заробітної плати у навчальному закладі. Спочатку гроші надходять на рахунок самого закладу, потім на рахунки підрозділів, потім конкретному співробітнику. Відповідно кожний підрозділ має відомість, що містить список його співробітників та розмір їх заробітної плати.

**Вибір шаблону**

Для розв’язання першої задачі був обраний структурний шаблон проектування «Декоратор» оскільки згідно з теоретичними відомостями він призначений для реалізування можливостей, які потрібні не всім об’єктам та

не завжди і можуть бути видалені при необхідності, що відповідає поставленій задачі, оскільки аналізуючи поставлену ситуацію було виявлено що деякі функції об’єктів є додатковими і присутні не у всіх об’єктах.

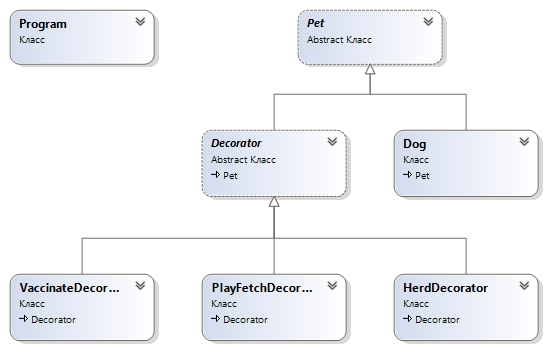
**Текст програми**

|  |
| --- |
| **Програмний код** |
| class Program  {  static void Main(string[] args)  {  Pet myFirstPet = new Dog("Molly", 18, 2, 5, "female", "chihuahua", "Ann Smith");  myFirstPet.ShowInfo();  Console.WriteLine("\n");  myFirstPet.DoActivity();  myFirstPet.MakeSound();  Console.WriteLine("-------------------");  Pet someDog = new Dog("Archie", 58, 32, 6, "male", "golden retriever", "Ann Smith");  Decorator goodBoy = new PlayFetchDecorator();  goodBoy.Decorate(someDog);  goodBoy.ShowInfo();  Console.WriteLine("\n");  goodBoy.DoActivity();  goodBoy.MakeSound();  Console.WriteLine("-------------------");  Pet newPet = new Dog("Ollie", 52, 17, 5, "male", "border collie", "Noah Johnson");  Decorator playfullDog = new PlayFetchDecorator();  playfullDog.Decorate(newPet);  Decorator workingDog = new HerdDecorator();  workingDog.Decorate(playfullDog);  Decorator vaccinatedDog = new VaccinateDecorator();  vaccinatedDog.Decorate(workingDog);  vaccinatedDog.ShowInfo();  Console.WriteLine("\n");  vaccinatedDog.DoActivity();  vaccinatedDog.MakeSound();  Console.WriteLine("-------------------");  Console.ReadKey();  }  }  abstract class Pet  {  protected string Name;  protected int Height;  protected int Weight;  protected int Age;  protected string Gender;  protected string Breed;  protected string Owner;  public abstract void ShowInfo();  public abstract void MakeSound();  public abstract void DoActivity();  }  class Dog : Pet  {  public Dog(string name, int height, int weight, int age, string gender, string breed, string owner)  {  this.Name = name;  this.Height = height;  this.Weight = weight;  this.Age = age;  this.Gender = gender;  this.Breed = breed;  this.Owner = owner;  }  public override void ShowInfo()  {  Console.WriteLine("This is "+this.Name+", a dog of "+this.Owner);  Console.WriteLine("It's a "+this.Gender+" "+this.Breed);  Console.WriteLine("height: " + this.Height+" cm"+" and weight: "+this.Weight +" kg");  Console.Write(this.Age+" years old");  }  public override void MakeSound()  {  Console.WriteLine("Bark");  }  public override void DoActivity()  {  Console.WriteLine("Making owner happy)");  }  }  abstract class Decorator:Pet  {  protected Pet DecoratedPet;  public void Decorate(Pet decoratedPet)  {  this.DecoratedPet = decoratedPet;  }  public override void ShowInfo()  {  if (DecoratedPet != null) DecoratedPet.ShowInfo();  }  public override void MakeSound()  {  if (DecoratedPet != null) DecoratedPet.MakeSound();  }  public override void DoActivity()  {  if (DecoratedPet != null) DecoratedPet.DoActivity();  }  }  class PlayFetchDecorator:Decorator  {  public override void ShowInfo()  {  base.ShowInfo();  Console.WriteLine("");  Console.Write("like playing fetch ");  }  public override void DoActivity()  {  base.DoActivity();  Console.WriteLine("Playing fetch");  }  }  class HerdDecorator : Decorator  {  public override void ShowInfo()  {  base.ShowInfo();  Console.WriteLine("");  Console.Write("shepherd ");  }  public override void DoActivity()  {  base.DoActivity();  Console.WriteLine("Herding livestock");  }  }  class VaccinateDecorator : Decorator  {  public override void ShowInfo()  {  base.ShowInfo();  Console.Write("vaccinated ");  }  }  } |

**Приклад результатів роботи програми**

|  |
| --- |
| **Вивід у консоль** |
|  |

**Діаграма класів**

****

**Вибір шаблону**

Для розв’язання другої задачі був обраний структурний шаблон проектування «Компонувальник» оскільки згідно з теоретичними відомостями він призначений для уніфікованої обробки деревовидної структури, що відповідає поставленій задачі, оскільки аналізуючи поставлену ситуацію було виявлено що для всіх підрозділів закладу виконуються однакові дії.

**Текст програми**

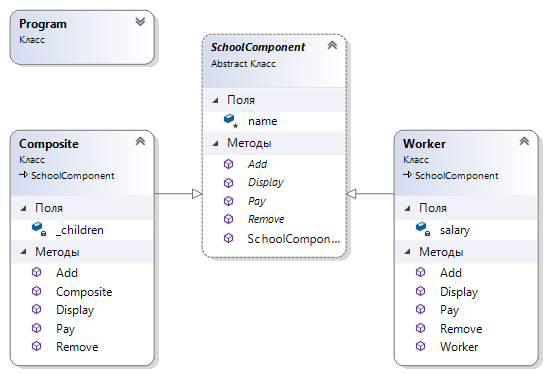
|  |
| --- |
| **Програмний код** |
| class Program  {  static void Main(string[] args)  {  Composite KPI = new Composite("KPI");  Composite FAM = new Composite("FAM");  Composite IPT = new Composite("IPT");  KPI.Add(FAM);  KPI.Add(IPT);  Worker lecturer = new Worker("Ann Smith", 3);  Worker proffesor = new Worker("Noah Johnson", 7);  Worker assistant = new Worker("James Williams", 4);  FAM.Add(lecturer);  FAM.Add(assistant);  IPT.Add(proffesor);  KPI.Display(2);  Console.WriteLine("");  int budget = 20;  int balance = KPI.Pay(budget);  Console.WriteLine("Budget to pay workers: " + budget);  if (balance>=0)  Console.WriteLine("Remains after payment: " + KPI.Pay(budget));  else Console.WriteLine("Not enough resources. Lacks " + Math.Abs(balance));  Console.ReadKey();  }  }  **abstract class SchoolComponent**  **{**  **protected string name;**  **public SchoolComponent(string name)**  **{**  **this.name = name;**  **}**  **public abstract void Add(SchoolComponent c);**  **public abstract void Remove(SchoolComponent c); public abstract int Pay(int budget); public abstract void Display(int depth);**  **}**  **class Composite:SchoolComponent**  **{**  **private List<SchoolComponent> \_children = new List<SchoolComponent>();**  **public Composite(string name):base(name)**  **{**  **}**  **public override void Add(SchoolComponent component)**  **{**  **\_children.Add(component);**  **}**  **public override void Remove(SchoolComponent component)**  **{**  **\_children.Remove(component);**  **}**  **public override int Pay(int budget)**  **{**  **foreach (SchoolComponent component in \_children)**  **budget = component.Pay(budget);**  **return budget;**  **}**  **public override void Display(int depth)**  **{**  **Console.WriteLine(new String('-', depth) + name);**  **// Recursively display child nodes**  **foreach (SchoolComponent component in \_children)**  **component.Display(depth + 2);**  **}**  **}**  **class Worker:SchoolComponent**  **{**  **int salary;**  **public Worker(string name, int salary):base(name)**  **{**  **this.salary = salary;**  **}**  **public override void Add(SchoolComponent c)**  **{**  **throw new NotImplementedException();**  **}**  **public override void Remove(SchoolComponent c)**  **{**  **throw new NotImplementedException();**  **}**  **public override int Pay(int budget)**  **{**  **return budget - this.salary;**  **}**  **public override void Display(int depth)**  **{**  **Console.WriteLine(new String('-', depth) + name + " salary: " + salary.ToString());**  **}**  **}** |

**Приклад результатів роботи програми**

|  |
| --- |
| **Вивід у консоль (перший випадок)** |
|  |

|  |
| --- |
| **Вивід у консоль (другий випадок)** |
|  |

**Діаграма класів**

****

**Висновки щодо роботи програм**

Виконавши дану лабораторну роботу було створено дві програми мовою C# що демонструють ознайомлення з використанням шаблонів проектування. Шляхом аналізу заданих ситуацій і порівняння їх характеристик з характеристиками стандартних зразків використання запропонованих шаблонів проектування для реалізації програм було обрано шаблони «Декоратор» та «Компонувальник» і показано доцільність цього вибору шляхом демонстрації роботи програм.

Розробка велася за допомогою Microsoft Visual Studio.