

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 “Об'єктно орієнтоване програмування”

тема “Структурні шаблони”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) II курсу  групи КП-01  Северин Юрій Юрійович  (*прізвище, ім’я, по батькові*)  Варіант № 14 | Київ 2021 | Перевірила  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

**Мета роботи**

Ознайомитися зі структурними шаблонами у C# та реалізувати дві задачі за допомогою цих паттернів.

**Постановка завдання**

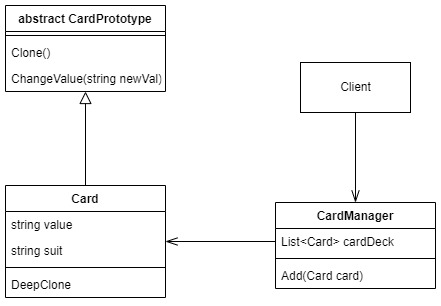
1. Гра у карти. За допомогою шаблону проєктування створити колоду карт. Для цього використати відповідні прототипи гральних карт кожної масті. У створених карт змінювати тільки їх ранг. На екран вивести список отриманих таким чином карт.
2. Компанія займається виробництвом корма для тварин, а саме для собак, котів та птахів. В асортименті компанії є корма преміум класу та бюджетні, що відрізняються якість інгредієнтів відповідно. За допомогою шаблону проєктування реалізувати процес випуску кормів всіх цінових категорій для всіх видів тварин.

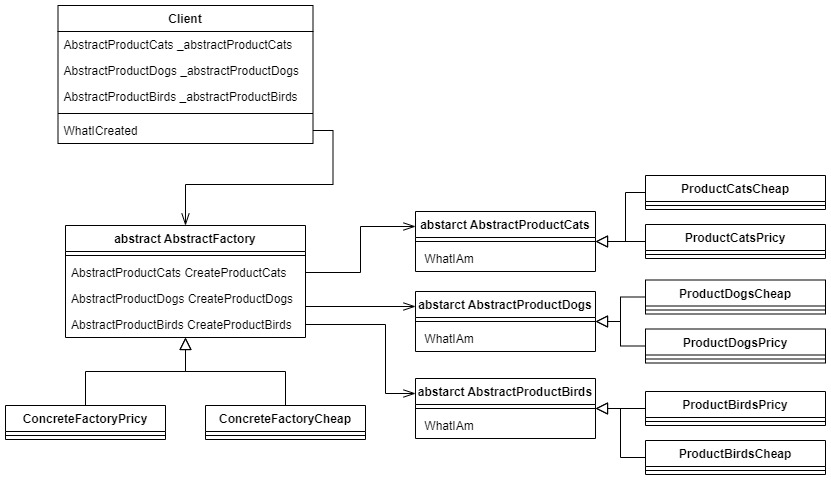
**Текст коду програми**

|  |
| --- |
| **1** |
| using System;  using System.Collections.Generic;  using System.IO;  using System.Runtime.Serialization.Formatters.Binary;  using System.Runtime.Serialization;  namespace Prototype\_Example2  {  class MainApp  {  static void Main()  {  CardManager cardManager = new CardManager();  List<string> values = new List<string>{ "7", "8", "9", "10", "J", "Q", "K", "A"};  Card clubs = new Card("6", "clubs");  Card diamonds = new Card("6", "diamonds");  Card hearts = new Card("6", "hearts");  Card spades = new Card("6", "spades");  cardManager.Add(clubs);  cardManager.Add(diamonds);  cardManager.Add(hearts);  cardManager.Add(spades);  foreach (var a in values)  {  cardManager.Add((clubs.Clone() as Card).ChangeValue(a));  cardManager.Add((diamonds.Clone() as Card).ChangeValue(a));  cardManager.Add((hearts.Clone() as Card).ChangeValue(a));  cardManager.Add((spades.Clone() as Card).ChangeValue(a));  }  foreach(var card in cardManager.cardDeck)  {  Console.WriteLine("{0, 8} | {1,1}", card.suit, card.value);  }  Console.ReadKey();  }  }  [Serializable]  abstract class CardPrototype  {  public abstract CardPrototype Clone();  public abstract Card ChangeValue(string newVal);  }  [Serializable]  class Card : CardPrototype  {  public string value;  public string suit;  public Card(string value, string suit)  {  this.value = value;  this.suit = suit;  }  public override Card ChangeValue(string newVal)  {  value = newVal;  return this;  }  public override CardPrototype Clone()  {  Console.WriteLine($"Cloning card of {suit}.");  return this.DeepClone() as CardPrototype;  }  public object DeepClone()  {  object clone = null;  using (MemoryStream tempStream = new MemoryStream())  {  BinaryFormatter binFormatter = new BinaryFormatter(null,  new StreamingContext(StreamingContextStates.Clone));  binFormatter.Serialize(tempStream, this);  tempStream.Seek(0, SeekOrigin.Begin);  clone = binFormatter.Deserialize(tempStream);  }  return clone;  }  }  class CardManager  {  public List<Card> cardDeck = new List<Card>();  public void Add(Card card)  {  cardDeck.Add(card);  }  }  } |

|  |
| --- |
| **2** |
| using System;  namespace task4  {  class MainApp  {  public static void Main()  {  // Abstract factory #1  Console.WriteLine("Creating pricy products");  AbstractFactory factoryPricy = new ConcreteFactoryPricy();  Client client1 = new Client(factoryPricy);  client1.WhatICreated();  Console.WriteLine("\nCreating cheap products");  // Abstract factory #2  AbstractFactory factorycheap = new ConcreteFactoryCheap();  Client client2 = new Client(factorycheap);  client2.WhatICreated();  // Wait for user input  Console.ReadKey();  }  }  class Client  {  private AbstractProductCats \_abstractProductCats;  private AbstractProductDogs \_abstractProductDogs;  private AbstractProductBirds \_abstractProductBirds;  public Client(AbstractFactory factory)  {  \_abstractProductCats = factory.CreateProductCats();  \_abstractProductDogs = factory.CreateProductDogs();  \_abstractProductBirds = factory.CreateProductBirds();  }  public void WhatICreated()  {  \_abstractProductCats.WhatIAm();  \_abstractProductDogs.WhatIAm();  \_abstractProductBirds.WhatIAm();  }  }  } |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace task4  {  abstract class AbstractFactory  {  public abstract AbstractProductCats CreateProductCats();  public abstract AbstractProductDogs CreateProductDogs();  public abstract AbstractProductBirds CreateProductBirds();  }  class ConcreteFactoryCheap : AbstractFactory  {  public override AbstractProductCats CreateProductCats()  {  return new ProductCatsCheap();  }  public override AbstractProductDogs CreateProductDogs()  {  return new ProductDogsCheap();  }  public override AbstractProductBirds CreateProductBirds()  {  return new ProductBirdsCheap();  }  }  class ConcreteFactoryPricy : AbstractFactory  {  public override AbstractProductCats CreateProductCats()  {  return new ProductCatsPricy();  }  public override AbstractProductDogs CreateProductDogs()  {  return new ProductDogsPricy();  }  public override AbstractProductBirds CreateProductBirds()  {  return new ProductBirdsPricy();  }  }  } |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace task4  {  #region Abstract products  abstract class AbstractProductCats  {  public abstract void WhatIAm();  }  abstract class AbstractProductDogs  {  public abstract void WhatIAm();  }  abstract class AbstractProductBirds  {  public abstract void WhatIAm();  }  #endregion  class ProductCatsCheap : AbstractProductCats  {  public override void WhatIAm()  {  Console.WriteLine( "Cheap cats products");  }  }  class ProductDogsCheap : AbstractProductDogs  {  public override void WhatIAm()  {  Console.WriteLine( "Cheap dogs products");  }  }  class ProductCatsPricy : AbstractProductCats  {  public override void WhatIAm()  {  Console.WriteLine( "Pricy cats products");  }  }  class ProductDogsPricy : AbstractProductDogs  {  public override void WhatIAm()  {  Console.WriteLine( "Pricy dogs products");  }  }  class ProductBirdsPricy : AbstractProductBirds  {  public override void WhatIAm()  {  Console.WriteLine( "Pricy birds products");  }  }  class ProductBirdsCheap : AbstractProductBirds  {  public override void WhatIAm()  {  Console.WriteLine( "Cheap birds products");  }  }  } |

**UML діаграма**

****



**Висновки**

Ознайомитилися зі структурними шаблонами у C# та реалізували дві задачі за допомогою цих паттернів.