МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Лабораторна робота №1

з дисципліни «Об'єктно оріентовне програмування»

Тема: «Шаблони проектування. Породжуючі шаблони»

Виконав:

студенти групи ІП-81

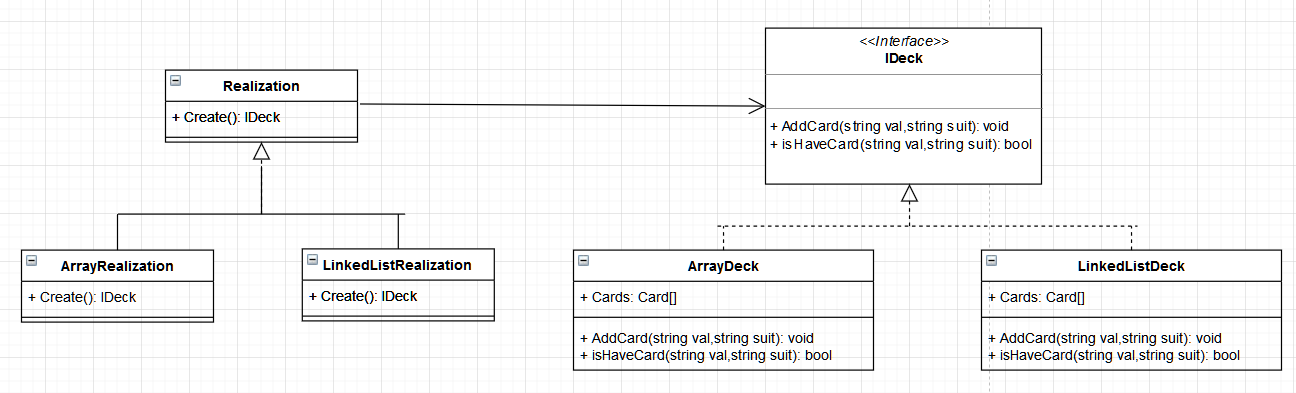
Балачін Петро

Київ-2020

**Мета роботи**: ознайомитися з основними шаблонами проектування, навчитися застосовувати їх при проектуванні і розробці ПЗ.

**Хід роботи:**

**Варіант 3**



class Program

{

class Program

{

static void Main(string[] args)

{

Realization real; //initialisation of Factory class

real = ChooseRealisation(); //picking a creator's type

IDeck deck = real.Create(); //Creating concrete product by

// general interface

AddCardLoop(deck); //Working with concrete product

}

/// <Summary>

/// Function to choose realisation between Array deck and Linked list deck.

/// </Summary>

public static Realization ChooseRealisation()

{

Console.WriteLine("What type of deck you want to create?");

Console.WriteLine("\ta - Array deck");

Console.WriteLine("\tl - Linked list deck");

switch (Console.ReadLine())

{

case "a":

return new ArrayRealization();

case "l":

return new LinkedListRealization();

default:

Console.WriteLine("There is no such option. Press any key to try again.");

Console.ReadKey();

Console.Clear();

return ChooseRealisation();

}

}

/// <Summary>

/// Function to add Card in Deck as long as user wants.

/// </Summary>

public static void AddCardLoop(IDeck deck)

{

Console.Clear();

Console.WriteLine(deck.ToString());

Console.WriteLine("Do you want to add new card to a deck?");

Console.WriteLine("\ty - Yes");

Console.WriteLine("\tn - No");

switch (Console.ReadLine())

{

case "y":

Console.Clear();

Console.WriteLine("Write the value of new card");

string value = Console.ReadLine();

Console.WriteLine("Write the suit of new card");

string suit = Console.ReadLine();

deck.AddCard(value, suit);

Console.WriteLine("Press any key to continue");

Console.ReadKey();

Console.Clear();

AddCardLoop(deck);

break;

case "n":

Console.Clear();

Console.WriteLine(deck.ToString());

Console.WriteLine("Press any key to quit");

Console.ReadKey();

Environment.Exit(0);

break;

default:

Console.WriteLine("There is no such option. Press any key to try again.");

Console.ReadKey();

Console.Clear();

AddCardLoop(deck);

break;

}

enum Suits

{

Hearts,Diamonds,Clubs,Spades

}

enum Values

{

Six,Seven,Eight,Nine,Ten,Jack,Queen,King,Ace

}

/// <Summary>

/// Class,objects of which represent's game cards with Suit and Value.

/// </Summary>

class Card

{

public Suits Suit { get; private set;}

public Values Value { get; private set; }

/// <Summary>

/// Constructor of card.

/// </Summary>

public Card(string suit,string value)

{

if (Enum.IsDefined(typeof(Values), value) && Enum.IsDefined(typeof(Suits), suit))

{

this.Suit = (Suits)Enum.Parse(typeof(Suits), suit);

this.Value = (Values)Enum.Parse(typeof(Values), value);

}

else

{

Console.WriteLine("Enter valid name of card");

}

}

/// <Summary>

/// Function for useful outputting in console.

/// </Summary>

public override string ToString()

{

return Value + " of " + Suit;

}

/// <Summary>

/// Function to compare objects of this type.

/// </Summary>

public bool Equals(Card card2)

{

return (Suit == card2.Suit

&& Value == card2.Value);

}

}

/// <Summary>

/// Class that represents Linked List.

/// </Summary>

public class LinkedList

{

private Node \_head;

/// <Summary>

/// Mehtod to add new element in Linked List

/// </Summary>

public void Add(Object data)

{

if (\_head == null)

{

\_head = new Node();

\_head.data = data;

}

else

{

Node thisNode = new Node();

thisNode.data = data;

Node last = \_head;

while(last.next != null)

{

last = last.next;

}

last.next = thisNode;

}

}

public override string ToString()

{

string output = "";

Node each = \_head;

while (each != null)

{

if(each.next == null)

{

output += each.data.ToString();

}

else

{

output += each.data.ToString() + ",";

}

each = each.next;

}

return output;

}

/// <Summary>

/// Function to check if Linked List is Empty

/// </Summary>

public bool isEmpty()

{

return \_head == null;

}

}

/// <Summary>

/// Class that represents element of the Linked List with the data and link

/// to next element.

/// </Summary>

internal class Node

{

public Node next;

public Object data;

}

abstract class Realization

{

/// <Summary>

/// Factory method creates general IDeck object.

/// </Summary>

abstract public IDeck Create();

}

/// <Summary>

/// Creator Class of Array-type Deck.

/// </Summary>

class ArrayRealization : Realization

{

public override IDeck Create()

{

return new ArrayDeck();

}

}

/// <Summary>

/// Creator Class of Linked List-type Deck.

/// </Summary>

class LinkedListRealization : Realization

{

public override IDeck Create()

{

return new LinkedListDeck();

/// <Summary>

/// General Interface for concrete products.

/// </Summary>

interface IDeck

{

/// <Summary>

/// Function to add a card to some type of deck.

/// </Summary>

void AddCard(string val, string suit);

/// <Summary>

/// Function to check if card is in the deck.

/// </Summary>

bool isHaveCard(string val, string suit);

/// <Summary>

/// Function for useful outputting in console.

/// </Summary>

/// <Summary>

/// One of concrete products,that represents deck based on array.

/// </Summary>

class ArrayDeck : IDeck

{

public Card[] Cards;

public ArrayDeck()

{

Cards = new Card[] { };

}

public bool isHaveCard(string val, string suit)

{

bool result = false;

for(int i = 0; i < Cards.Length; i++)

{

if (Cards[i].Equals(new Card(suit, val)))

result = true;

}

return result;

}

public void AddCard(string val, string suit)

{

if (Enum.IsDefined(typeof(Values), val) && Enum.IsDefined(typeof(Suits), suit))

{

if (isHaveCard(val, suit))

{

Console.WriteLine("This card has been added yet.");

}

else

{

Array.Resize(ref Cards, Cards.Length + 1);

Cards[Cards.Length - 1] = new Card(suit, val);

}

}

else

{

Console.WriteLine("Enter valid name of card");

}

}

public override string ToString()

{

string output = "Array deck is :";

if (Cards.Length > 0)

{

for (int i = 0; i < Cards.Length; i++)

{

output += "\n\t" + Cards[i].ToString();

}

}

else

{

output += " Empty";

}

return output;

}

/// <Summary>

/// One of concrete products,that represents deck based on Linked List.

/// </Summary>

class LinkedListDeck : IDeck

{

public LinkedList Cards;

public LinkedListDeck()

{

Cards = new LinkedList();

Console.WriteLine("LinkedListDeck has created.");

}

public bool isHaveCard(string val, string suit)

{

bool result = false;

string[] cards = Cards.ToString().Split(",");

for (int i = 0; i < cards.Length; i++)

{

if (cards[i] == new Card(suit, val).ToString())

result = true;

}

return result;

}

public void AddCard(string val, string suit)

{

if (Enum.IsDefined(typeof(Values), val) && Enum.IsDefined(typeof(Suits), suit))

{

if(isHaveCard(val,suit))

{

Console.WriteLine("This card has been added yet.");

}

else

{

Cards.Add(new Card(suit, val));

}

}

else

{

Console.WriteLine("Enter valid name of card");

}

}

public override string ToString()

{

string output = "Linked List deck is :";

string[] cards = Cards.ToString().Split(",");

if (!Cards.isEmpty())

{

for (int i = 0; i < cards.Length; i++)

{

output += "\n\t" + cards[i];

}

}

else

{

output += "Empty";

}

return output;

}

**Висновок:**Я засвоїв теоретичний матеріал з відповідної теми та закріпив його,виконавши практичні завдання.