Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки

інформації та управління

**ЗВІТ**

з комп’ютерного практикуму № 3

на тему :

„ Шаблоны проектирования. Структурные шаблоны”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав студент** |  | *ІП-63 Зозуля А.В.* |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові ) |  |  |
|  |  |  |  |  |
| **Прийняв** |  |  |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2018

**ЗМІСТ**

[**1. Мета роботи**](#_oodamo6mklpc) **1**

[**2. Постановка задачі**](#_17dp8vu) **3**

[**3. UML ДІАГРАМА КЛАСУ**](#_3rdcrjn) **4**

[**4. Висновок**](#_lnxbz9) **5**

[**5. Текст програмного коду**](#_35nkun2) **5**

# 1. Мета роботи

*Мета роботи* – ознакомиться с основными шаблонами проектирования, научиться применять их при проектировании и разработке ПО.

# 

# 

# 

# 2. Постановка задачі

Постановка задачи к компьютерному практикуму № 3

При выполнении компьютерного практикума следует реализовать следующие действия:

1) Реализовать задачу согласно вариантов, приведенных ниже (можно

предложить реализацию собственной задачи)

2) Кратко описать архитектуру проекта

3) Привести UML-диаграмму классов

4) Описать используемые поведенческие паттерны. Для каждого паттерна

нужно указать составные классы и их назначение

7) Заказ на получение гранта для обучения может находиться в нескольких

состояниях: создан, рассматривается, отложен, отклонен, подтвержден,

отозван и т. д. Определить логику изменения состояний и разработать

модель системы.

# 3. UML ДІАГРАМА КЛАСУ

# 

# 4. Висновок

В ході виконання лабораторної роботи було здійснено розробку програмного забезпечення, використовуючи поведінковий паттерн State .

# 5. Текст програмного коду

using System;

namespace lab3

{

public class Grant

{

private IState \_state = new Created();

public void SetState(IState newState)

{

\_state = newState;

}

public void PrintState()

{

Console.WriteLine("Current state : " + \_state.GetState());

}

public void ChangeState()

{

PrintState();

var index = 0;

var states = \_state.GetNextState();

if (states == null)

{

Console.WriteLine("You can't change grant state.");

}

else

{

foreach (var state in states)

{

Console.WriteLine("#" + (index + 1) + " change to " + state.GetState());

index++;

}

do

{

index = 0;

Console.Write("Choose the index of next state to change : ");

index = Convert.ToInt32(Console.ReadLine());

} while (index > states.Count || index < 0);

SetState(states[index - 1]);

}

}

}

}

using System;

using System.Collections.Generic;

namespace lab3

{

public interface IState

{

string GetState();

List<IState> GetNextState();

}

public class Created : IState

{

public string GetState()

{

return "created";

}

public List<IState> GetNextState()

{

var list = new List<IState>();

list.Add(new Postponed());

list.Add(new Reviewed());

return list;

}

}

public class Reviewed : IState

{

public string GetState()

{

return "reviewed";

}

public List<IState> GetNextState()

{

var list = new List<IState>();

list.Add(new Postponed());

list.Add(new Confirmed());

list.Add(new Rejected());

return list;

}

}

public class Postponed : IState

{

public string GetState()

{

return "postponed";

}

public List<IState> GetNextState()

{

var list = new List<IState>();

list.Add(new Reviewed());

list.Add(new Confirmed());

list.Add(new Rejected());

return list;

}

}

public class Rejected : IState

{

public string GetState()

{

return "rejected";

}

public List<IState> GetNextState()

{

return null;

}

}

public class Confirmed : IState

{

public string GetState()

{

return "confirmed";

}

public List<IState> GetNextState()

{

var list = new List<IState>();

list.Add(new Recalled());

return list;

}

}

public class Recalled : IState

{

public string GetState()

{

return "recalled";

}

public List<IState> GetNextState()

{

var list = new List<IState>();

list.Add(new Returned());

return list;

}

}

public class Returned : IState

{

public string GetState()

{

return "returned";

}

public List<IState> GetNextState()

{

var list = new List<IState>();

list.Add(new Recalled());

return list;

}

}

}

using System;

namespace lab3

{

class Program

{

static void Main(string[] args)

{

var grant = new Grant();

var answer = "start";

do

{

grant.PrintState();

do

{

Console.Write("Do you want to change grant state?(y/n): ");

answer = Console.ReadLine();

} while (answer != "Y" && answer != "y" && answer != "N" && answer != "n");

if (answer == "Y" || answer == "y")

{

grant.ChangeState();

}

} while (answer == "Y" || answer == "y");

}

}

}