

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 №** 2\_\_

з дисципліни “Об’єктно орієнтоване програмування ”

тема “**Реалізація складних структурних шаблонів проектування**”

|  |  |  |
| --- | --- | --- |
| Виконала  студентка II курсу  групи КП-82  Лахман Ксенія Вікторівна  (*прізвище, ім’я, по батькові*) |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Ознайомлення з основними характеристиками шаблонів «Заступник», «Фасад», «Адаптер» та «Легковаговик», запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

**Постановка задачі:**

1. Розробити модуль програмної системи обліку абонентів оператора мобільного зв’язку, який при спробі абонента перейти на інший тариф буде запитувати в нього дані щодо PUK-коду телефону і відображати перелік доступних тарифів тільки у разі коректного введення цього коду.

2. За допомогою шаблона проектування забезпечити виведення на шахову дошку фігур в їх початковому розташуванні.

**Обгрунтування вибору шаблонів:**

1. Заступник

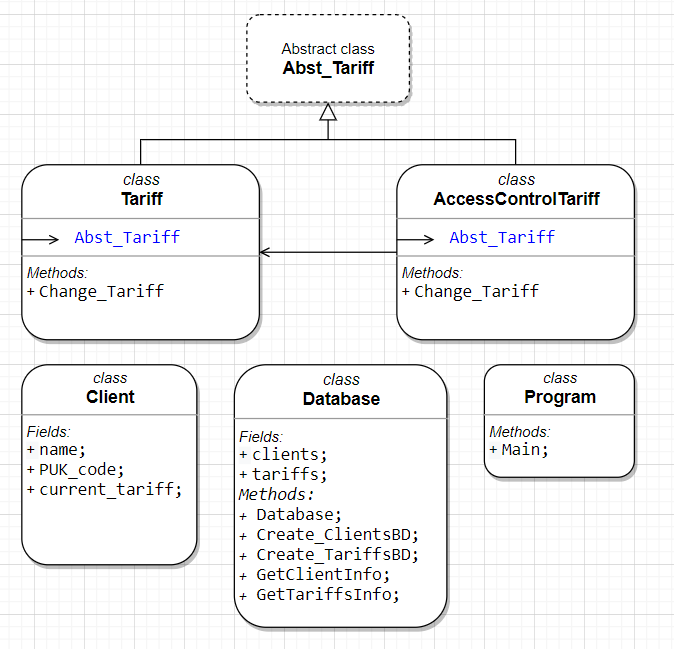
* Створення нового тарифу - дорога операція
* Заступник для аутентифікації (заступник-захисник): перевіряє правильність умов доступу до об’єкту;

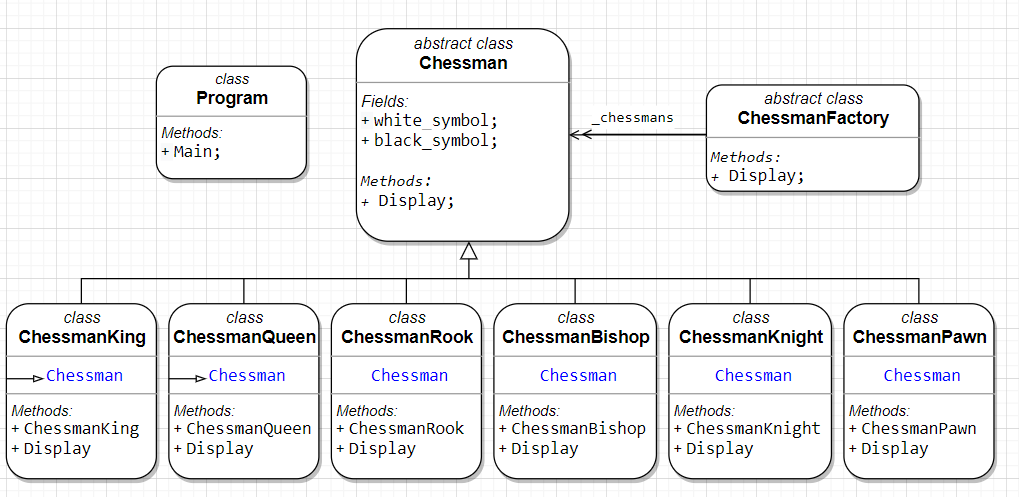
1. Легковагковик

* Використовується велика кількість подібних об'єктів - 16 пішаків, по 4 слона, коня, тур і тд
* Ідентичність кожного об’єкту не має значення

**Діаграма класів**

**Задача1**

****

**Задача2**

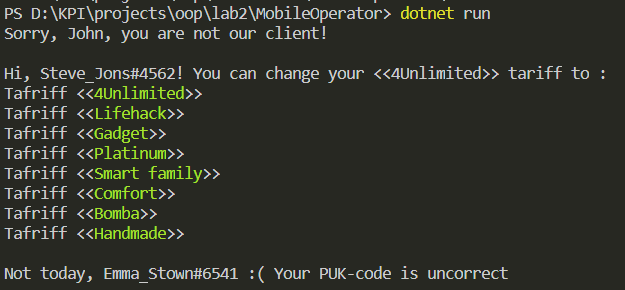
**Тексти коду програм**

|  |
| --- |
| **Задача 1** |
| namespace MobileOperator  {  class Program  {  static void Main(string[] args)  {  AccessControlTariff Prime\_ControlTariff = new AccessControlTariff();  Client m1 = new Client() { name = "John", PUK\_code = 12345678, current\_tariff = "Lifehack"};  Client m2 = new Client() { name = "Steve\_Jons#4562", PUK\_code = 11994523, current\_tariff = "4Unlimited"};  Client m3 = new Client() { name = "Emma\_Stown#6541", PUK\_code = 12345678, current\_tariff = "Lifehack"};    Prime\_ControlTariff.Change\_Tariff(m1);  Console.WriteLine("");  Prime\_ControlTariff.Change\_Tariff(m2);  Console.WriteLine("");    Prime\_ControlTariff.Change\_Tariff(m3);  Console.WriteLine("");  // Console.ReadKey();  }  }  class Client  {  public string name;  public int PUK\_code;  public string current\_tariff;  }  abstract class Abst\_Tariff // interface  {  public abstract void Change\_Tariff(Client m);  }  class Tariff : Abst\_Tariff // RealSubject  {  Database bd = new Database();  public override void Change\_Tariff(Client m)  {  Console.WriteLine("Hi, {0}! You can change your <<{1}>> tariff to :", m.name, m.current\_tariff);  bd.GetTariffsInfo();  }  }  public class Database  {  private Hashtable clients = new Hashtable();  private List<string> tariffs = new List<string>();  public Database()  {  Create\_ClientsBD();  Create\_TariffsBD();  }  void Create\_ClientsBD()  {  this.clients.Add("Kris\_Evans#8542", 13212312);  this.clients.Add("Steve\_Jons#4562", 11994523);  this.clients.Add("Emma\_Stown#6541", 15263524);  this.clients.Add("Jennifer\_Aniston#6378", 15974826);  this.clients.Add("Johnny\_Depp#4523", 96300548);  this.clients.Add("Angelina\_Jolie#7420", 41203269);  this.clients.Add("Brooke\_Shields#3097", 78030614);  }  void Create\_TariffsBD()  {  tariffs.Add("4Unlimited");  tariffs.Add("Lifehack");  tariffs.Add("Gadget");  tariffs.Add("Platinum");  tariffs.Add("Smart family");  tariffs.Add("Comfort");  tariffs.Add("Bomba");  tariffs.Add("Handmade");  }  public int GetClientInfo(string name)  {  try {  return (int)this.clients[name];  }  catch {  return -1;  }  }  public void GetTariffsInfo()  {  foreach(string tr in tariffs)  {  Console.Write("Tafriff <<");  Console.ForegroundColor = ConsoleColor.Green;  Console.Write(tr);  Console.ResetColor();  Console.WriteLine(">>");  }  }  }  class AccessControlTariff : Abst\_Tariff // Proxy  {  Tariff tariff = new Tariff();  Database bd = new Database();  public override void Change\_Tariff(Client cl)  {  int client\_pass = bd.GetClientInfo(cl.name);  if (client\_pass == cl.PUK\_code)  {  tariff.Change\_Tariff(cl);  }  else if (client\_pass == -1)  {  Console.WriteLine("Sorry, {0}, you are not our client!", cl.name);  }  else  Console.WriteLine("Not today, {0} :( Your PUK-code is uncorrect", cl.name);  }  }  } |

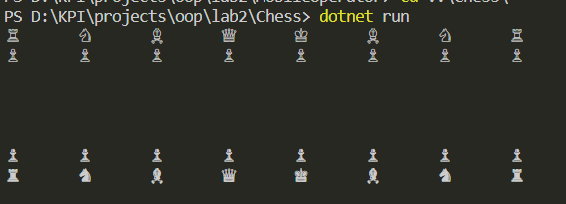
|  |
| --- |
| **Задача 2** |
| namespace Chess  {  class Program  {  static void Main()  {  string[] main\_chessmans = {"Rook", "Knight", "Bishop", "Queen", "King", "Bishop", "Knight", "Rook"};  string[] pawn\_chessmans = {"Pawn", "Pawn", "Pawn", "Pawn", "Pawn", "Pawn", "Pawn", "Pawn"};  ChessmanFactory factory = new ChessmanFactory();    //white\_player  bool state = true;  foreach (string chman in main\_chessmans) {  Chessman character = factory.GetCharacter(chman);  character.Display(state);  }  Console.WriteLine("");  foreach (string chman in pawn\_chessmans) {  Chessman character = factory.GetCharacter(chman);  character.Display(state);  }  Console.WriteLine("\n\n\n\n");  //black\_player  state = false;  foreach (string chman in pawn\_chessmans) {  Chessman character = factory.GetCharacter(chman);  character.Display(state);  }  Console.WriteLine("");  foreach (string chman in main\_chessmans) {  Chessman character = factory.GetCharacter(chman);  character.Display(state);  }  Console.WriteLine("");    Console.ReadKey();  }  }  class ChessmanFactory // The 'FlyweightFactory' class  {  private Dictionary<string, Chessman> \_chessmans = new Dictionary<string, Chessman>();  public Chessman GetCharacter(string key)  {  // Uses "lazy initialization"  Chessman chessman = null;  if (\_chessmans.ContainsKey(key)) {  chessman = \_chessmans[key];  } else {  switch (key)  {  case "King": chessman = new ChessmanKing(); break;  case "Queen": chessman = new ChessmanQueen(); break;  case "Rook": chessman = new ChessmanRook(); break;  case "Bishop": chessman = new ChessmanBishop(); break;  case "Knight": chessman = new ChessmanKnight(); break;  case "Pawn": chessman = new ChessmanPawn(); break;  }  \_chessmans.Add(key, chessman);  }  return chessman;  }  }  // The 'Flyweight' abstract class  abstract class Chessman  {  protected string white\_symbol;  protected string black\_symbol;  // protected bool state; // true - white, false - black  public abstract void Display(bool state);  }  // A 'ConcreteFlyweight' class  class ChessmanKing : Chessman  {  public ChessmanKing()  {  this.white\_symbol = "\x2654";  this.black\_symbol = "\x265A";  }  public override void Display(bool state)  {  Console.OutputEncoding = System.Text.Encoding.Unicode;  if (state) {  Console.Write(this.white\_symbol + '\t');  } else {  Console.Write(this.black\_symbol + '\t');  }  }  }  /// A 'ConcreteFlyweight' class  class ChessmanQueen : Chessman  {  public ChessmanQueen()  {  this.white\_symbol = "\x2655";  this.black\_symbol = "\x265B";  }  public override void Display(bool state)  {  Console.OutputEncoding = System.Text.Encoding.Unicode;  if (state) {  Console.Write(this.white\_symbol + '\t');  } else {  Console.Write(this.black\_symbol + '\t');  }  }  }    // A 'ConcreteFlyweight' classes  class ChessmanRook : Chessman  {  public ChessmanRook()  {  this.white\_symbol = "\x2656";  this.black\_symbol = "\x265C";  }  public override void Display(bool state)  {  Console.OutputEncoding = System.Text.Encoding.Unicode;  if (state) {  Console.Write(this.white\_symbol + '\t');  } else {  Console.Write(this.black\_symbol + '\t');  }  }  }  class ChessmanBishop : Chessman  {  public ChessmanBishop ()  {  this.white\_symbol = "\x2657";  this.black\_symbol = "\x265D";  }  public override void Display(bool state)  {  Console.OutputEncoding = System.Text.Encoding.Unicode;  if (state) {  Console.Write(this.white\_symbol + '\t');  } else {  Console.Write(this.black\_symbol + '\t');  }  }  }  class ChessmanKnight : Chessman  {  public ChessmanKnight ()  {  this.white\_symbol = "\x2658";  this.black\_symbol = "\x265E";  }  public override void Display(bool state)  {  Console.OutputEncoding = System.Text.Encoding.Unicode;  if (state) {  Console.Write(this.white\_symbol + '\t');  } else {  Console.Write(this.black\_symbol + '\t');  }  }  }  class ChessmanPawn : Chessman  {  public ChessmanPawn ()  {  this.white\_symbol = "\x2659";  this.black\_symbol = "\x265F";  }  public override void Display(bool state)  {  Console.OutputEncoding = System.Text.Encoding.Unicode;  if (state) {  Console.Write(this.white\_symbol + '\t');  } else {  Console.Write(this.black\_symbol + '\t');  }  }  }  } |

**Приклади роботи програми:**

**Задача1**



**Задача2**



**Висновки**

Виконавши дану лабораторну роботу, я ознайомилася з основними характеристиками шаблонів «Заступник» та «Легковаговик». Я запам’ятала ситуації, коли використання цих шаблонів є доцільним, вивчила назви шахматних фігур англійською мовою, зрозуміла нюанси написання коду з використанням шаблонів.