

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 №**

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**

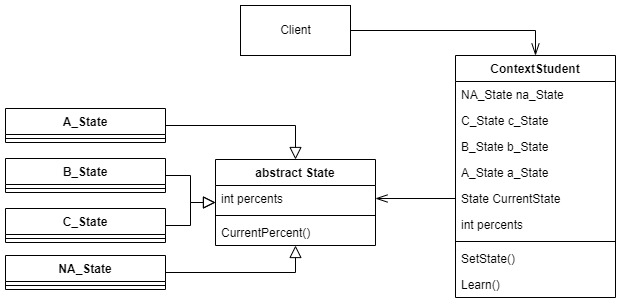
1. За допомогою шаблону проєктування відтворити процес вивчення студентом певного предмету та здачі ним екзамену. За кожне самостійне заняття студент опановує 5% матеріалу. При опануванні студентом більше 60% матеріалу, він переходить у категорію потенційних «трійочників», при опануванні більше 80% - у категорію студентів, що претендують на «четвірку», та при опануванні більше 95% матеріалу – стає потенційним відмінником.
2. Для формування команди спеціалістів на новий проект компанія звертається до рекрутингової агенції та надає інформацію щодо вимог до кандидатів на вакантні посади. Рекрутингова агенція, в свою чергу, має певну базу спеціалістів з інформацією про їх професійні якості та бажані позиції, з якої і вибирає кандидатів на задані посади. Рекрутингова агенція може звертатися до  іншої подібної агенції з метою проведення пошуку кандидатів чи вакансій в базі даних останньої. Іноді спеціалісти за власною ініціативою відправляють відомості про себе до банка резюме рекрутингової компанії, щоб та повідомляла про нові вакансії, які можуть бути для них цікавими. За допомогою шаблону проєктування змоделювати процес взаємодії фахівців та компаній-замовників через рекрутингову агенцію.

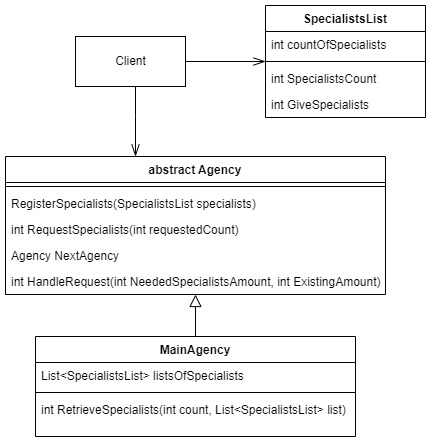
**Текст коду програми**

|  |
| --- |
| **1** |
| using System;  namespace lab5  {  class Program  {  public static void Main(string[] args)  {  ContextStudent contextStudent = new ContextStudent();  for (int i = 0; i < 25; i++)  {  contextStudent.Learn();  }  Console.ReadKey();  }  }  } |
| using System;  namespace lab5  {  class ContextStudent  {  NA\_State na\_State = new NA\_State();  С\_State с\_State = new С\_State();  B\_State b\_State = new B\_State();  A\_State a\_State = new A\_State();  private State CurrentState;  int percents = 0;  public ContextStudent()  {  CurrentState = na\_State;  }  void SetState()  {  if (percents >= na\_State.percents)  {  CurrentState = na\_State;  }  if (percents >= с\_State.percents)  {  CurrentState = с\_State;  }  if (percents >= b\_State.percents)  {  CurrentState = b\_State;  }  if (percents >= a\_State.percents)  {  CurrentState = a\_State;  }  }  public void Learn()  {  if (percents < 95)  {  percents += 5;  SetState();  CurrentState.CurrentPercent();  }  else  {  Console.WriteLine("Learned to maximum");  }  }  }  } |
| using System;  namespace lab5  {  public abstract class State  {  public int percents { get; private protected set; }  abstract public void CurrentPercent();  }  class A\_State : State  {  public A\_State()  {  percents = 95;  }  override public void CurrentPercent()  {  Console.WriteLine("I have >95 points. I`m learning");  }  }  class B\_State : State  {  public B\_State()  {  percents = 80;  }  override public void CurrentPercent()  {  Console.WriteLine("I have 80-95 points. I`m learning");  }  }  class С\_State : State  {  public С\_State()  {  percents = 60;  }  override public void CurrentPercent()  {  Console.WriteLine("I have 60-80 points. I`m learning");  }  }  class NA\_State : State  {  public NA\_State()  {  percents = 0;  }  override public void CurrentPercent()  {  Console.WriteLine("I have big problems. I`m learning");  }  }  } |

|  |
| --- |
| **2** |
| namespace task6  {  abstract class Agency  {  public abstract void RegisterSpecialists(SpecialistsList specialists);  public abstract int RequestSpecialists(int requestedCount);    public Agency NextAgency { get; set; }  public abstract int HandleRequest(int NeededSpecialistsAmount, int ExistingAmount);  }  } |
| using System.Collections.Generic;  using System;  namespace task6  {  class MainAgency : Agency  {  private List<SpecialistsList> listsOfSpecialists;  public MainAgency()  {  listsOfSpecialists = new List<SpecialistsList>();  }  public override int HandleRequest(int NeededSpecialistsAmount, int ExistingAmount)  {  int a = RequestSpecialists(NeededSpecialistsAmount);  ExistingAmount += a;  NeededSpecialistsAmount -= a;  if (NeededSpecialistsAmount <= 0)  {  Console.WriteLine("Hired all specialists");  return ExistingAmount;  }  else if(NextAgency != null)  {  Console.WriteLine("Going to next Agency");  ExistingAmount += NextAgency.HandleRequest(NeededSpecialistsAmount - ExistingAmount, ExistingAmount);  }  else  {  Console.WriteLine("There are not enought specialists");  }  return ExistingAmount;  }  public override void RegisterSpecialists(SpecialistsList specialists)  {  if (!listsOfSpecialists.Contains(specialists))  {  listsOfSpecialists.Add(specialists);  }  }  public override int RequestSpecialists(int requestedCount)  {  List<SpecialistsList> currentMarketSpecialists = new List<SpecialistsList>();  int guarantedCount = 0;  foreach (SpecialistsList spec in listsOfSpecialists)  {  guarantedCount += spec.SpecialistsCount;  currentMarketSpecialists.Add(spec);  if (guarantedCount >= requestedCount)  {  guarantedCount = requestedCount;  break;  }  }  int receivedCount = RetrieveSpecialists(guarantedCount, currentMarketSpecialists);  return receivedCount;  }  private int RetrieveSpecialists(int count, List<SpecialistsList> list)  {  int receivedCount = 0;  foreach (SpecialistsList spec in list)  {  int givenSpecs = spec.GiveSpecialists(count);  receivedCount += givenSpecs;    count -= givenSpecs;  }  return receivedCount;  }  }  } |
| using System;  namespace task6  {  class Program  {  static void Main(string[] args)  {  SpecialistsList man1 = new SpecialistsList(100);  SpecialistsList man2 = new SpecialistsList(50);  SpecialistsList man3 = new SpecialistsList(70);  Agency sup = new MainAgency();  sup.RegisterSpecialists(man1);  Agency sup2 = new MainAgency();  sup.NextAgency = sup2;  sup2.RegisterSpecialists(man2);  sup2.RegisterSpecialists(man3);  HireSpecs(sup, 300);  Console.ReadKey();  }  static void HireSpecs(Agency \_agency, int count)  {  int specialistsCount = 0;  if (\_agency != null)  {  Console.WriteLine($"Requesting {count} specialists in agency");  int receivedCount = \_agency.HandleRequest(count, 0);  specialistsCount += receivedCount;  Console.WriteLine($"Agency has provided {receivedCount} specialists");  }  }  }  } |
| namespace task6  {  class SpecialistsList  {  int countOfSpecialists;  public SpecialistsList(int countOfSpecialists)  {  this.countOfSpecialists = countOfSpecialists;  }  public int SpecialistsCount  {  get  {  return countOfSpecialists;  }  set  {  if (countOfSpecialists >= 0)  {  countOfSpecialists = value;  }  }  }  public int GiveSpecialists(int requestedCount)  {  int giveCount = 0;  if (countOfSpecialists < requestedCount)  {  giveCount = countOfSpecialists;    countOfSpecialists = 0;  }  else  {  giveCount = requestedCount;  countOfSpecialists -= requestedCount;  }  return giveCount;  }  }  } |

**UML діаграма**





**Висновки**

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