НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КПІ»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

ЛАБОРАТОРНА РОБОТА №2

*з дисципліни*

*«Об’єктно-орієнтоване програмування»*

ТЕМА: «Реалізація складних структурних шаблонів

проектування»

Підготував: студент групи КП-51

Бабенко Валерій Павлович

Перевірила:

Заболотня Тетяна Миколаївна

Київ 2017

# 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

Варіант 2

# В системі дистанційної освіти реалізувати механізм, який буде

# дозволяти користувачу он-лайн курсу виконувати завдання з поточного

# модуля (генерувати список завдань з наявного загального масиву завдань)

# тільки за умови наявності виконаних завдань (позитивних оцінок) з

# минулих модулів.

# Стара система оцінювання знань студентів характеризувалась

# такими рисами: 100-бальна шкала, до 60 балів – незадовільно; 60-69 –

# трійка, 70-84 – четвірка; 85-100 – п’ятірка. Оцінка при цьому ставилась

# цифрою (2, 3, 4, 5). Нова система оцінювання має 60-бальну систему та

# вимагає проставлення оцінки прописом. Нова шкала балів змінена

# пропорційно старої шкали. За допомогою шаблона проектування

# забезпечити виведення оцінки студента на основі суми набраних ним балів

# як у старій, та і у новій системі оцінок.

# Вибір структурного шаблону програмування

1. «Тільки за умови наявності» - ознака «**Проксі**»-захисника.
2. Стара і нова системи. Це «**Адаптер**».

# Лістинг програми

|  |
| --- |
| Program.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OOP2\_Lab2\_New  {  class Program  {  static void Main(string[] args)  {  #region Marks  MarkTransformer mt1 = new MarkTransformerNewSystem();  mt1.transform(58);  MarkTransformer mt2 = new MarkTransformer();  mt2.transform(87);  #endregion  #region Testing  Student valera = new Student("Valera");  AbstractModule am1 = new Module();  am1.passTheModule(valera, 1);  am1.passTheModule(valera, 3);  valera.display();  AbstractModule am2 = new ProtectedModule();  am2.passTheModule(valera, 4);  am2.passTheModule(valera, 2);  valera.display();  #endregion  }  }  } |

|  |
| --- |
| MarkTransformer.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OOP2\_Lab2\_New  {  // Target; from 100 to 2-5  class MarkTransformer  {  public virtual void transform(int points100)  {  int mark;  if (points100 < 60)  {  mark = 2;  }  else if (points100 <= 69)  {  mark = 3;  }  else if (points100 <= 84)  {  mark = 4;  }  else  {  mark = 5;  }  Console.WriteLine("Mark: "+mark);  }  }  // Adaptee; from 60 to String  class MarkStringTransformer  {  public void getHandwritten(int points60)  {  string handwrittenmark;  if (points60 < 36)  {  handwrittenmark = "two";  }  else if (points60 <= 41)  {  handwrittenmark = "three";  }  else if (points60 <= 50)  {  handwrittenmark = "four";  }  else if (points60 <= 60)  {  handwrittenmark = "five";  }  else  {  throw new Exception();  }  Console.WriteLine("Mark: "+handwrittenmark);  }  public int get100from60(int points60)  {  return (int)((points60 / 60) \* 100);  }  }  //Adapter  class MarkTransformerNewSystem : MarkTransformer  {  public override void transform(int points60)  {  MarkStringTransformer mst = new MarkStringTransformer();  mst.getHandwritten(points60);  }  }  } |

|  |
| --- |
| Testing.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OOP2\_Lab2\_New  {  class Student  {  private string name;  public string Name  {  get { return name; }  }  private int[] marks;  public int[] Marks  {  get { return marks; }  }  public Student(string name)  {  this.name = name;  marks = new int[4];  for (int i=0; i<4; i++)  {  marks[i] = 0;  }  }  public bool setMark(int module, int mark)  {  if (mark > 10 || mark < 0)  {  return false;  }  try  {  this.marks[(module - 1)] = mark;  }  catch (IndexOutOfRangeException e)  {  Console.WriteLine(e.Message);  throw new ArgumentOutOfRangeException("Index parameter is out of range!", e);  }  return true;  }  public void display()  {  Console.WriteLine("Name: {0}", name);  for(int i=0; i<4; i++)  {  if (this.marks[i] != 0)  {  Console.WriteLine("{0} module - {1} points", (i+1), marks[i]);  }  }  }  }  abstract class AbstractModule  {  abstract public void passTheModule(Student student, int module);  }  class Module : AbstractModule  {  public override void passTheModule(Student student, int module)  {  Random random = new Random();  int mark = random.Next(0, 10);  student.setMark(module, mark);  if (mark < (10 \* 0.6))  {  Console.WriteLine("{0}, today, you didn't pass {1} module", student.Name, module);  }  else  {  Console.WriteLine("{0}, today, you passed {1} module", student.Name, module);  }  }  }  class ProtectedModule : AbstractModule  {  Module moduleClass = new Module();  public override void passTheModule(Student student, int module)  {  int[] arr = student.Marks;  for (int i=0; i<(module-1); i++)  {  if (arr[i] < (10 \* 0.6))  {  Console.WriteLine("You didn't pass {0} module!", (i+1));  return;  }  }  moduleClass.passTheModule(student, module);  }  }  } |

