НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КПІ»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

ЛАБОРАТОРНА РОБОТА №1

*з дисципліни*

*«Об’єктно-орієнтоване програмування»*

ТЕМА: «Реалізація структурних шаблонів проектування»

Підготував: студент групи КП-51

Бабенко Валерій Павлович

Перевірила:

Заболотня Тетяна Миколаївна

Київ 2016

# Мета роботи

# Ознайомлення з основними характеристиками шаблонів «Декоратор», «Компонувальник» та «Міст», 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

1. Магазин «Все для дому» реалізує господарські товари. Всі товари мають спільні характеристики: наприклад, назва, ціна, відділ, де вони продаються, термін придатності чи гарантійний термін, а також оригінальні характеристики, які залежать від виду товару. Одні товари продаються тільки вроздріб, а інші можуть бути продані й оптом (при цьому ціна товару стає на 10% меншою). Створити базовий клас «Товар» та похідні класи, що міститимуть додаткову інформацію про товари, які можуть бути проданими як вроздріб, так і оптом.
2. Розробити програмне забезпечення автоматичної зйомки фотоапаратом. В залежності від освітлення та часу виконувати підключення тих чи інших можливостей фотоапарату: спалаху, знищення ефекту червоних очей, автофокус на посмішці, знімання об’єкту, що рухається тощо.

# Вибір структурного шаблону програмування

1. Всі товари мають оригінальні характеристики. «(..)а також ОРИГІНАЛЬНІ характеристики (..)» - такі надбудови – ознака «**Декоратора**».
2. В ЗАЛЕЖНОСТІ від освітлення та часу виконувати підключення тих чи інших можливостей. Різні реалізації з однією абстракцією – це шаблон «**Міст**».

# Лістинг програми

|  |
| --- |
| Program.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OOP2\_Lab1  {  class Program  {  static void Main(string[] args)  {  Console.Write("Camera test: (type any key)");  Console.ReadKey();  Console.WriteLine();  #region Camera test  Camera cam = new Camera();  cam.takePhoto(LightingLevel.Low, FlashButton.AutoSet);  cam.takePhoto(LightingLevel.High, FlashButton.On);  cam.takePhoto(LightingLevel.High, FlashButton.Off);  cam.takePhoto(LightingLevel.High, FlashButton.AutoSet);  cam.takePhoto(LightingLevel.Medium, FlashButton.AutoSet);  Console.ReadKey();  #endregion  Console.Write("\nEconomic merchandise test: (type any key)");  Console.ReadKey();  Console.WriteLine("\n");  #region Merchandise test  EconomicMerchandise em1 = new EconomicMerchandise("TitaniumHole", 13.2, Sections.A, 17);  Decorator D1 = new BucketDecorator(0.5, 0.6, 0.4);  D1.setMerchandiseType(em1);  D1.viewInfo();  int d1amount = 14;  double d1price = D1.buy(d1amount);  Console.Write("Amount: " + d1amount + "\nPaid price: " + d1price + "\n\n");  EconomicMerchandise em2 = new EconomicMerchandise("SoftSkinnyRag", 13, Sections.B, 1);  Decorator D2 = new SpongeDecorator(0.5, 0.6, 0.4);  D2.setMerchandiseType(em2);  D2.viewInfo();  int d2amount = 10;  double d2price = D2.buy(d2amount);  Console.Write("Amount: " + d2amount + "\nPaid price: " + d2price + "\n\n");  #endregion  }  }  } |

|  |
| --- |
| Camera.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OOP2\_Lab1  {  enum LightingLevel { Low, Medium, High };  enum FlashButton { Off, On, AutoSet };  class Camera{  List<Photo> photos;  public Camera()  {  photos = new List<Photo>();  }  public void takePhoto(LightingLevel ll, FlashButton fb)  {  AbstractCameraMode mode = getConfigs(ll, fb);  Photo photo = mode.takeAPhoto();  photos.Add(photo);  }  private AbstractCameraMode getConfigs(LightingLevel lightingLevel, FlashButton flashbutt)  {  if (flashbutt == FlashButton.On)  return new AbstractCameraMode(new LightingLevelCameraModeLow());  else if (flashbutt == FlashButton.Off)  return new AbstractCameraMode(new LightingLevelCameraModeHigh());  else  switch (lightingLevel)  {  case LightingLevel.Low:  return new AbstractCameraMode(new LightingLevelCameraModeLow());  case LightingLevel.Medium:  case LightingLevel.High:  return new AbstractCameraMode(new LightingLevelCameraModeHigh());  default: return null;  }  }  }  public class Photo  {  private int buffer;  private bool lit;  public Photo()  {  lit = false;  }  public void setLitProp()  {  //there's photo lights  lit = true;  }  }  interface ICameraMode  {  Photo takeAPhoto();  }  class AbstractCameraMode  {  ICameraMode camera;  public AbstractCameraMode(ICameraMode cam)  {  this.camera = cam;  }  public Photo takeAPhoto()  {  return camera.takeAPhoto();  }  }  class LightingLevelCameraModeHigh : ICameraMode  {  public Photo takeAPhoto()  {  Photo photo = new Photo();  Console.WriteLine("Got a photo. Flash wasn't activated");  return photo;  }  }  class LightingLevelCameraModeLow : ICameraMode  {  public Photo takeAPhoto()  {  Photo photo = new Photo();  Console.WriteLine("Got a photo. Flash was activated");  photo.setLitProp();  return photo;  }  }  } |

|  |
| --- |
| Goods.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OOP2\_Lab1  {  enum Sections  {  A, B, C  }  abstract class Merchandise  {  protected string nameOfModel;  protected double price;  protected Sections section;  protected int guaranteePeriodInMonths;  public abstract double buy(int amount);  public abstract void viewInfo();  }  class EconomicMerchandise : Merchandise  {  public EconomicMerchandise(string nameOfModel, double price, Sections section, int guaranteePeriodInMonths)  {  this.nameOfModel = nameOfModel;  this.price = price;  this.section = section;  this.guaranteePeriodInMonths = guaranteePeriodInMonths;  }  public override double buy(int amount)  {  return price \* amount;  }  public override void viewInfo()  {  Console.Write("Model name: " + nameOfModel + "\n" +  "Price: " + price + "\n" +  "Section: " + section + "\n" +  "Guarantee period (in month): " + guaranteePeriodInMonths + "\n");  }  }  abstract class Decorator : Merchandise  {  protected Merchandise merch;  public void setMerchandiseType(Merchandise baseMerchandise)  {  merch = baseMerchandise;  }  public override double buy(int amount)  {  if (merch != null)  {  return merch.buy(amount);  }  else  {  throw new Exception();  }  }  public override void viewInfo()  {  if (merch != null)  {  merch.viewInfo();  }  }  }  class BucketDecorator : Decorator  {  private double sale = 0.1;  private double radiusInMetersLower;  private double radiusInMetersUpper;  private double height = 0;  public BucketDecorator(double radiusInMetersLower, double radiusInMetersUpper, double height)  {  this.radiusInMetersLower = radiusInMetersLower;  this.radiusInMetersUpper = radiusInMetersUpper;  this.height = height;  }  public override double buy(int amount)  {  return base.buy(amount) \* (1 - sale);  }  public override void viewInfo()  {  base.viewInfo();  Console.Write("Sale: " + (sale\*100) + "%\n" +  "Lower radius of bucket (in meters) " + radiusInMetersLower + "\n" +  "Upper radius of bucket (in meters) " + radiusInMetersUpper + "\n" +  "Height (in meters): " + height + "\n" +  "Volume: " + getVolume() + "\n");  }  private double getVolume()  {  return (Math.PI \* height \* (Math.Pow(radiusInMetersLower, 2) + radiusInMetersLower \* radiusInMetersUpper + Math.Pow(radiusInMetersUpper, 2)) / 3);  }  }  class SpongeDecorator : Decorator  {  private string colour = "yellow";  private double thickness;  private double height;  private double width;  public SpongeDecorator(double height, double width, double thickness)  {  this.thickness = thickness;  this.height = height;  this.width = width;  }  public override void viewInfo()  {  base.viewInfo();  Console.Write("Colour: " + colour + "\n" +  "Height (in meters) " + height + "\n" +  "Width (in meters) " + width + "\n" +  "Thickness (in meters): " + thickness + "\n" +  "Volume: " + getVolume() + "\n");  }  private double getVolume()  {  return height \* width \* thickness;  }  }  } |

**Діаграма класів**

