НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КПІ»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

ЛАБОРАТОРНА РОБОТА №3

*з дисципліни*

*«Об’єктно-орієнтоване програмування»*

ТЕМА: «Реалізація породжуючих шаблонів проектування»

Підготував: студент групи КП-51

Бабенко Валерій Павлович

Перевірила:

Заболотня Тетяна Миколаївна

Київ 2017

# Мета роботи

# Ознайомлення з основними характеристиками всіх

# породжуючих шаблонів («Прототипу», «Фабричного методу»,

# «Абстрактної фабрики», «Одинака», «Будівельника»), 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

# За допомогою шаблона проектування реалізувати процес

# випуску книжок за заготовленими раніше матрицями. Кожна книга

# характеризується іменем автора, видавництвом, де вона була надрукована,

# серією та номером видання. Масив сторінок є тим об’єктом, який

# необхідно розтиражувати для кожної книги.

# За допомогою шаблона «Абстрактна фабрика» реалізувати

# віртуальні меблеві фабрики. Кожна фабрика виготовляє вітальні, м’які

# меблі, кухні. Характеризуються фабрики матеріалами, які

# використовуються для виготовлення меблів, та вартістю продукції. В

# залежності від побажань та фінансових можливостей замовника обирати ту

# фабрику, яка буде створювати для нього меблі в даний момент часу

# Вибір структурного шаблону програмування

1. Розтиражування – ознака «**Прототипу**»
2. «**Абстрактна фабрика**» з умови.

# Лістинг програми

|  |
| --- |
| Program.cs |
| namespace OOP2\_Lab3  {  class Program  {  static void Main(string[] args)  {  #region Furniture  Buyer valera = new Buyer("Valera");  Food keep = new Food();  keep.getInfo();  valera.Money = 10000;  Furniture.getCosts();  Furniture furniture = new Furniture();  furniture.createOrder(valera, "SiciliaFactory");  furniture.createOrder(valera, null);  furniture.executeOrder();  furniture.freezeFood(keep);  keep.getInfo();  #endregion  Console.WriteLine(" ");  #region Replication  ProductCache.loadCache();  Book clonedBook1 = (Book)ProductCache.getBook("B1");  clonedBook1.getInfo();  Book clonedBook2 = (Book)ProductCache.getBook("B1");  clonedBook2.getInfo();  clonedBook1 = (Book)ProductCache.getBook("B2");  clonedBook1.getInfo();  #endregion  }  }  } |

|  |
| --- |
| Furniture.cs |
| namespace OOP2\_Lab3  {    abstract class FurnitureFactory  {  public abstract LivingRoomFurniture createLivingRoomFurniture();  public abstract KitchenFurniture createKitchenFurniture();  public abstract UpholsteredFurniture createUpholsteredFurniture();  }  class ValeronManFactory : FurnitureFactory  {  public static int getCost()  {  return 10000;  }  public override LivingRoomFurniture createLivingRoomFurniture()  {  return new Closet();  }  public override KitchenFurniture createKitchenFurniture()  {  return new PlasticRefrigerator();  }  public override UpholsteredFurniture createUpholsteredFurniture()  {  return new WoodenCouch();  }  }  class SiciliaFactory : FurnitureFactory  {  public static int getCost()  {  return 30000;  }  public override LivingRoomFurniture createLivingRoomFurniture()  {  return new MirrorCloset();  }  public override KitchenFurniture createKitchenFurniture()  {  return new MetalRefrigerator();  }  public override UpholsteredFurniture createUpholsteredFurniture()  {  return new Sofa();  }  }  abstract class LivingRoomFurniture  {  }  abstract class KitchenFurniture  {  public abstract void freezeFood(Food keep);  }  abstract class UpholsteredFurniture  {    }  class Closet : LivingRoomFurniture  {  }  class MirrorCloset : LivingRoomFurniture  {  }  class PlasticRefrigerator : KitchenFurniture  {  public override void freezeFood(Food keep)  {  Console.WriteLine("Waiting 5 seconds...");  Thread.Sleep(5000);  keep.freeze();  }  }  class MetalRefrigerator : KitchenFurniture  {  public override void freezeFood(Food keep)  {  Console.WriteLine("Waiting 2 seconds...");  Thread.Sleep(2000);  keep.freeze();  }  }  class WoodenCouch : UpholsteredFurniture  {  }  class Sofa : UpholsteredFurniture  {  }  class Furniture  {  private LivingRoomFurniture livRoomFur;  private KitchenFurniture kitFur;  private UpholsteredFurniture uphFur;  FurnitureFactory factory;  public void executeOrder()  {  if (factory != null)  {  livRoomFur = factory.createLivingRoomFurniture();  kitFur = factory.createKitchenFurniture();  uphFur = factory.createUpholsteredFurniture();  }  else  {  Console.WriteLine("Unknown factory!");  }  }  public static void getCosts()  {  Console.WriteLine("ValeronManFactory - " + ValeronManFactory.getCost() + "$");  Console.WriteLine("SiciliaFactory - " + SiciliaFactory.getCost() + "$");  }  public void freezeFood(Food keep)  {  if (kitFur != null)  {  kitFur.freezeFood(keep);  }  else  {  Console.WriteLine("Unknown kitchen!");  }  }  }  } |

|  |
| --- |
| Replication.cs |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OOP2\_Lab3  {  class Page  {  public int pageNumber;  public string text;  }  public class Book : ICloneable  {  private String SKU;  private String name;  private String author;  private int numberOfPages;  private Page[] pages;  private string issueSeries;  private string publishingHouse;  private int issueNumber;  public Object Clone()  {  Object clone = null;  try  {  clone = this.MemberwiseClone();  }  catch (Exception e)  {  Console.WriteLine(e.Message);  }  return clone;  }  public String getName()  {  return name;  }  public String getSKU()  {  return SKU;  }  public void setSKU(String \_sku)  {  SKU = \_sku;  }  public void setName(String name)  {  this.name = name;  }  public String getAuthor()  {  return author;  }  public void setAuthor(String author)  {  this.author = author;  }  public void getPages()  {  for (int i = 0; i < numberOfPages; i++)  {  Console.WriteLine(pages[i].text);  }  Console.WriteLine(" ");  }  public void setPages(string [] pagesArr)  {  int length = pagesArr.Length;  if (length <= 0)  {  throw new Exception();  }  this.numberOfPages = length;  pages = new Page[length];  for (int i = 0; i< length; i++)  {  pages[i] = new Page();  pages[i].pageNumber = i + 1;  pages[i].text = pagesArr[i];  }  }  public String getIssueSeries()  {  return issueSeries;  }  public int getIssueNumber()  {  return issueNumber;  }  public void setSeriesAndNumberOfIssue(String series, int number)  {  this.issueNumber = number;  this.issueSeries = series;  }  public string getPublishingHouse()  {  return publishingHouse;  }  public void setPublishingHouse(string house)  {  publishingHouse = house;  }  public void getInfo()  {  Console.WriteLine("Name: "+name+"; Author: "+author);  Console.WriteLine("Publishing house: " + publishingHouse + "; Series: " + issueSeries+"; Issue number: "+issueNumber+";");  Console.WriteLine("Number of pages: " + numberOfPages + "; Text:\n");  this.getPages();  }  }  public class ProductCache  {  private static Hashtable bookMap = new Hashtable();  public static Book getBook(String bookCode)  {  Book cachedBook = (Book)bookMap[bookCode];  return (Book)cachedBook.Clone();  }  public static void loadCache()  {  Book b1 = new Book();  string [] text1 = { "hi", "bye", "i'm good" };  b1.setPages(text1);  b1.setName("Oliver Twist");  b1.setAuthor("Charles Dickens");  b1.setSKU("B1");  b1.setPublishingHouse("Valerooon-man");  b1.setSeriesAndNumberOfIssue("sd89", 89);  bookMap[b1.getSKU()] = b1;  Book b2 = new Book();  string [] text2 = { "damn", "i'm", "good" };  b2.setPages(text2);  b2.setName("The Sea Wolf");  b2.setAuthor("Jack London");  b2.setSKU("B2");  b2.setPublishingHouse("Valerooon-man");  b2.setSeriesAndNumberOfIssue("hfhfhhg", 112);  bookMap[b2.getSKU()] = b2;  Book b3 = new Book();  string [] text3 = { "my", "life" };  b3.setPages(text3);  b3.setName("The call of the wild");  b3.setAuthor("Jack London");  b3.setSKU("B3");  b3.setPublishingHouse("Valerooon-man");  b3.setSeriesAndNumberOfIssue("ere34", 11);  bookMap[b3.getSKU()] = b3;  }  }  } |



