v

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 №**3

з дисципліни “Основи об’єктно-орієнтованого програмування”

тема “ Реалізація породжуючих шаблонів проектування ”

|  |  |  |
| --- | --- | --- |
| Виконала  студентка II курсу  групи КП-83  Дзьоба Уляна Олегівна  (*прізвище, ім’я, по батькові*)  варіант №5 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2020 р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2020

**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 завдання**

1. Всі користувачі одного комп’ютера працюють з одним і тим самим з’єднанням з Інтернетом. Це з’єднання зберігається одна з характеристик кожного користувача. За допомогою шаблона проектування забезпечити створення нових користувачів комп’ютера шляхом клонування базового екземпляра профілю користувача, при чому Інтернет з’єднання повинне залишатися єдиним об’єктом, без копій.
2. За допомогою шаблона «Абстрактна фабрика» організувати випуск пакетів соку з різних фруктів та ягід декількома заводами. Заводи відрізняються сортами сировини, упаковкою та ціною продукції. В залежності від пори року клієнти замовляють сік у різних заводів.

**Вибір шаблону**

Для розв’язання першої задачі був обраний породжуючий шаблон проектування «Прототип» оскільки це єдиний шаблон з серії породжуючих, котрий для створення нових об’єктів використовує клонування, а саме про клонування йдеться в умові задачі; також додатково використаний породжуючий шаблон “Одинак” для забезпечення унікальності об’єкту.

**Текст програми**

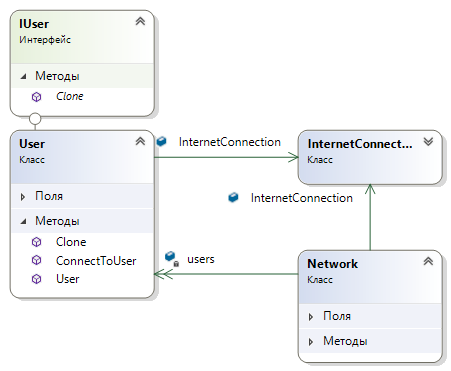
|  |
| --- |
| **Програмний код** |
| class Program  {  static void Main(string[] args)  {  InternetConnection internetConnection = InternetConnection.getInstance("Lanet", 35.7, 5);  List<User> administrationUsers = new List<User>();  User admin = new User(null, "127.0.0.1", "Admin", "admin@gmail.com", internetConnection);  User moderator = new User(null, "127.0.0.1", "Moderator", "moderator@gmail.com", internetConnection);  administrationUsers.Add(admin);  administrationUsers.Add(moderator);  Network network = new Network(internetConnection, administrationUsers);  network.AddUser("Ann Smith", "annsmith@gmail.com");  network.AddUser("Noah Johnson", "noahjohnson@gmail.com");  internetConnection.Provider = "Volia";  internetConnection.Speed = 30.1;  network.ShowInfo();  Console.ReadKey();  }  }  class InternetConnection  {  private static InternetConnection instance;  public string Provider;  public double Speed;  public int Ping;  protected InternetConnection(string provider, double speed, int ping)  {  Provider = provider;  Speed = speed;  Ping = ping;  }  public static InternetConnection getInstance(string provider, double speed, int ping)  {  if (instance == null)  instance = new InternetConnection(provider, speed, ping);  return instance;  }  }  interface IUser  {  User Clone();  void ConnectToUser(User contact);  }  class User:IUser  {  public InternetConnection InternetConnection;  public string IP;  public string Name;  public string Email;  public List<User> connectedUsers;  public User(List<User> connectedUsers, string ip, string name, string email, InternetConnection internetConnection)  {  IP = ip;  Name = name;  Email = email;  this.connectedUsers = connectedUsers;  InternetConnection = internetConnection;  }  public User Clone()  {  User clonedUser= (User)this.MemberwiseClone();  List<User> clonedConnectedUsers = new List<User>(this.connectedUsers);  return clonedUser;  }  public void ConnectToUser(User contact)  {  connectedUsers.Add(contact);  }  }  class Network  {  public InternetConnection InternetConnection;  public User templateUser;  public List<User> AdministrationUsers;  private List<User> users = new List<User>();  public Network(InternetConnection internetConnection, List<User> administrationUsers)  {  InternetConnection = internetConnection;  AdministrationUsers = administrationUsers;  templateUser= new User(AdministrationUsers, "192.168.0.1", null, null, InternetConnection);  }  public void AddUser(string name, string email)  {  User newUser = templateUser.Clone();  newUser.Name = name;  newUser.Email = email;  users.Add(newUser);  }  public void ShowInfo()  {  Console.WriteLine("Users in network:\n");  foreach(User user in users)  {  Console.WriteLine(user.Name);  Console.WriteLine(user.Email);  Console.WriteLine(user.IP);  Console.WriteLine("Internet provider: "+user.InternetConnection.Provider);  Console.WriteLine("Internet speed: " + user.InternetConnection.Speed);  Console.WriteLine("Connected to "+user.Name+":");  foreach(User connected in user.connectedUsers)  {  Console.WriteLine(connected.Name);  }  Console.WriteLine("\n");  }  }  } |

**Приклад результатів роботи програми**

|  |
| --- |
| **Вивід у консоль 1** |
|  |

|  |
| --- |
| **Вивід у консоль 2** |
|  |

**Діаграма класів**

****

**Вибір шаблону**

Для розв’язання другої задачі був обраний породжуючий шаблон проектування «Абстрактна фабрика» оскільки він заданий в умові завдання.

**Текст програми**

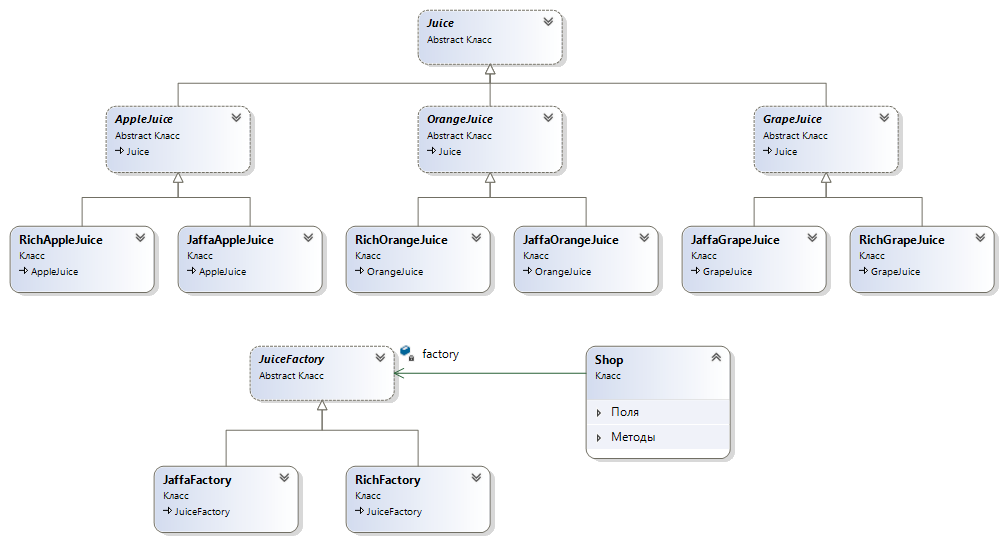
|  |
| --- |
| **Програмний код** |
| **class Program**  **{**  **static void Main(string[] args)**  **{**  **DateTime.season = Season.spring;**  **Shop shop = new Shop();**  **shop.ShowJuices();**  **Console.ReadKey();**  **}**  **}**    **abstract class JuiceFactory**  **{**  **public abstract AppleJuice MakeAppleJuice();**  **public abstract GrapeJuice MakeGrapeJuice();**  **public abstract OrangeJuice MakeOrangeJuice();**  **}**  **class JaffaFactory : JuiceFactory**  **{**  **public override AppleJuice MakeAppleJuice()**  **{**  **return new JaffaAppleJuice();**  **}**  **public override GrapeJuice MakeGrapeJuice()**  **{**  **return new JaffaGrapeJuice();**  **}**  **public override OrangeJuice MakeOrangeJuice()**  **{**  **return new JaffaOrangeJuice();**  **}**  **}**  **class RichFactory : JuiceFactory**  **{**  **public override AppleJuice MakeAppleJuice()**  **{**  **return new RichAppleJuice();**  **}**  **public override GrapeJuice MakeGrapeJuice()**  **{**  **return new RichGrapeJuice();**  **}**  **public override OrangeJuice MakeOrangeJuice()**  **{**  **return new RichOrangeJuice();**  **}**  **}**  **class Shop**  **{**    **private AppleJuice appleJuice;**  **private GrapeJuice grapeJuice;**  **private OrangeJuice orangeJuice;**  **JuiceFactory factory;**  **private void setFactory()**  **{**  **if(DateTime.season==Season.autumn|| DateTime.season == Season.winter)**  **{**  **factory = new RichFactory();**  **}**  **else factory = new JaffaFactory();**  **}**  **public Shop()**  **{**  **setFactory();**  **appleJuice = factory.MakeAppleJuice();**  **grapeJuice = factory.MakeGrapeJuice();**  **orangeJuice = factory.MakeOrangeJuice();**  **}**  **public void ShowJuices()**  **{**  **Console.WriteLine("In stock now:");**  **appleJuice.Show();**  **grapeJuice.Show();**  **orangeJuice.Show();**  **}**  **}**  **abstract class Juice**  **{**  **internal double price;**  **internal string package;**  **internal double sugar;**  **abstract public void Show();**  **}**  **abstract class AppleJuice:Juice**  **{**  **internal string variety;**  **public override void Show()**  **{**  **System.Console.WriteLine("from {0} apples", variety);**  **System.Console.WriteLine("sugar: {0} grams", sugar);**  **System.Console.WriteLine("in " + package);**  **System.Console.WriteLine(price + "$");**  **}**  **}**  **abstract class OrangeJuice:Juice**  **{**  **internal double vitaminC;**  **public override void Show()**  **{**  **System.Console.WriteLine("vitamin C: {0} grams", vitaminC);**  **System.Console.WriteLine("sugar: {0} grams", sugar);**  **System.Console.WriteLine("in " + package);**  **System.Console.WriteLine(price + "$");**  **}**  **}**    **abstract class GrapeJuice:Juice**  **{**  **internal string color;**  **public override void Show()**  **{**  **System.Console.WriteLine("from {0} grapes", color);**  **System.Console.WriteLine("sugar: {0} grams", sugar);**  **System.Console.WriteLine("in " + package);**  **System.Console.WriteLine(price + "$");**  **}**  **}**  **internal class JaffaAppleJuice : AppleJuice**  **{**  **public JaffaAppleJuice()**  **{**  **price = 3.2;**  **package = "PET plastic bottle";**  **sugar = 28.2;**  **variety = "Golden Delicious";**  **}**  **public override void Show()**  **{**  **System.Console.WriteLine("Jaffa Apple Juice");**  **base.Show();**  **System.Console.WriteLine("");**  **}**  **}**  **internal class RichAppleJuice : AppleJuice**  **{**  **public RichAppleJuice()**  **{**  **price = 3.5;**  **package = "tetra pack carton";**  **sugar = 28;**  **variety = "Granny Smith";**  **}**  **public override void Show()**  **{**  **System.Console.WriteLine("Rich Apple Juice");**  **base.Show();**  **System.Console.WriteLine("");**  **}**  **}** |

**Приклад результатів роботи програми**

|  |
| --- |
| **Використання першої фабрики** |
|  |

|  |
| --- |
| **Використання другої фабрики** |
|  |

**Діаграма класів**

****

**Висновки щодо роботи програм**

Виконавши дану лабораторну роботу було створено дві програми мовою C# що демонструють ознайомлення з використанням породжуючих шаблонів проектування. Шляхом аналізу заданих ситуацій і порівняння їх характеристик з характеристиками стандартних зразків використання запропонованих шаблонів проектування для реалізації програм було обрано шаблони «Заступник» та «Абстрактна фабрика» і показано доцільність цього вибору шляхом демонстрації роботи програм.

Розробка велася за допомогою Microsoft Visual Studio.