

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 №** 5

з дисципліни “ Об'єктно-орієнтоване програмування та шаблони проєктування ”

тема “Поведінкові шаблони проєктування”

|  |  |  |
| --- | --- | --- |
| Виконав  Студент II курсу  групи КП-02  Щербатюк Ярослав Сергійович  (*прізвище, ім’я, по батькові*)  варіант №21 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2021

**Мета роботи**

Навчитися на практиці визначати, який поведінковий шаблон проєктування слід використовувати в залежності від умов задачі.

**Постановка завдання**

1. За допомогою шаблону проєктування відтворити процес споживання певного продукту. Даний продукт має визначений термін зберігання. У випадку перевищення терміну зберігання, продукт стає шкідливим для здоров’я людини. У випадку використання продукту, коли термін його зберігання ще не сплив (тобто, коли продукт є корисним), загрози здоров’ю споживача немає.
2. За допомогою шаблону проєктування реалізувати алгоритм визначення купюр для видачі замовленої суми в банкоматі. У випадку, якщо не всю бажану суму можна видати великими купюрами (наприклад, по 200 грн.), залишок, який є менше 200 грн., буде виданий купюрами меншого номіналу. У випадку, якщо неможливо визначити необхідну комбінацію купюр, банкомат видає про це відповідне повідомлення та просить перевизначити іншу суму з урахуванням наявних купюр в банкоматі.

**UML-діаграми**

|  |
| --- |
| **Задача 1** |
|  |

|  |
| --- |
| **Задача 2** |
|  |

**Тексти коду програми**

|  |
| --- |
| **State.cs (1 задача)** |
| using System;  public interface IFoodState  {      void EatFood(Product p);  }  public class HealthyFoodState : IFoodState  {      public void EatFood(Product p)      {          if(p.ExpirationDate > p.DateOfUse)          {              Console.WriteLine($"Human eats {p.Name}");          }          else          {              p.State = new UnHealthyFoodState();              Console.WriteLine($"Human do not eat {p.Name}. {p.Name} is spoiled");          }          p.DateOfUse = p.DateOfUse.AddDays(1);      }  }  public class UnHealthyFoodState : IFoodState  {      public void EatFood(Product p)      {          Console.WriteLine($"Human do not eat {p.Name}. {p.Name} is spoiled");          p.DateOfUse = p.DateOfUse.AddDays(1);      }  }  public class Product  {      public DateTime ExpirationDate {get; private set;}      public string Name {get; private set;}      public IFoodState State {get; set;}      public DateTime DateOfUse {get; set;}      public Product(string name, DateTime expDate)      {          this.Name = name;          this.ExpirationDate = expDate;          this.DateOfUse = DateTime.Now;          State = new HealthyFoodState();      }      public void Eat()      {          State.EatFood(this);      }  } |

|  |
| --- |
| **ChainOfResponsisbility.cs (2 задача)** |
| using System.Collections.Generic;  using System;  public abstract class BankNoteHandler  {      protected BankNoteHandler sucsessor;      protected int bankNoteValue;      public BankNoteHandler(int bankNoteValue)      {          this.bankNoteValue = bankNoteValue;      }      public void SetSucsessor(BankNoteHandler h)      {          this.sucsessor = h;      }      public abstract void GetBankNotes(double amount, Dictionary<int, int> bnDict);      protected abstract int TakeBankNotes(ref double amount);  }  public class ConcreteBankNoteHandler : BankNoteHandler  {      public ConcreteBankNoteHandler(int bankNoteValue) : base(bankNoteValue) {}      public override void GetBankNotes(double amount, Dictionary<int, int> bnDict)      {          if(amount > 0 && this.sucsessor == null)          {              throw new System.Exception("Unable to give cash, try enter another amount");          }          int  takedBankNotes = TakeBankNotes(ref amount);          bnDict[this.bankNoteValue] = takedBankNotes;          if(amount > 0)              this.sucsessor.GetBankNotes(amount, bnDict);      }      protected override int TakeBankNotes(ref double amount)      {          int takedBankNotes = 0;          while(amount >= this.bankNoteValue)          {              amount -= this.bankNoteValue;              takedBankNotes ++;          }          return takedBankNotes;      }  }  public class ATM  {      private BankNoteHandler handler;      private Dictionary<int, int> bnDict;      public ATM()      {          BankNoteHandler bankNote1000 = new ConcreteBankNoteHandler(1000);          BankNoteHandler bankNote500 = new ConcreteBankNoteHandler(500);          BankNoteHandler bankNote200 = new ConcreteBankNoteHandler(200);          BankNoteHandler bankNote100 = new ConcreteBankNoteHandler(100);          BankNoteHandler bankNote50 = new ConcreteBankNoteHandler(50);          BankNoteHandler bankNote20 = new ConcreteBankNoteHandler(20);          BankNoteHandler bankNote10 = new ConcreteBankNoteHandler(10);          BankNoteHandler bankNote5 = new ConcreteBankNoteHandler(5);          BankNoteHandler bankNote2 = new ConcreteBankNoteHandler(2);          BankNoteHandler bankNote1 = new ConcreteBankNoteHandler(1);          bankNote1000.SetSucsessor(bankNote500);          bankNote500.SetSucsessor(bankNote200);          bankNote200.SetSucsessor(bankNote100);          bankNote100.SetSucsessor(bankNote50);          bankNote50.SetSucsessor(bankNote20);          bankNote20.SetSucsessor(bankNote10);          bankNote10.SetSucsessor(bankNote5);          bankNote5.SetSucsessor(bankNote2);          bankNote2.SetSucsessor(bankNote1);          this.FillDict();          this.handler = bankNote1000;      }      private void FillDict()      {          if (this.bnDict == null)              this.bnDict = new Dictionary<int, int>();            this.bnDict[1] = 0;          this.bnDict[2] = 0;          this.bnDict[5] = 0;          this.bnDict[10] = 0;          this.bnDict[50] = 0;          this.bnDict[100] = 0;          this.bnDict[200] = 0;          this.bnDict[500] = 0;          this.bnDict[1000] = 0;      }      private void PrintDict()      {          System.Console.WriteLine("Cash:");          foreach (KeyValuePair<int, int>  pair in this.bnDict)          {              if(pair.Value != 0)                  System.Console.WriteLine($"{pair.Key}x{pair.Value}");          }      }      public void GiveCash(double amount)      {          if (amount < 0)              throw new Exception("Amount can not be negative");          try          {              this.handler.GetBankNotes(amount, this.bnDict);              this.PrintDict();              this.FillDict();          }          catch (Exception ex)          {              throw ex;          }      }  } |

**Приклади роботи програм**

|  |
| --- |
| **Задача 1** |
| PS C:\Users\Yarik Scherbatyuk\Documents\OOP\_2course\_scherbatyukk\lab5\src\1> dotnet run  Enter commmand  createProduct Candy 3  Enter command(product mode)  eat  Human eats Candy  Enter command(product mode)  eat  Human eats Candy  Enter command(product mode)  eat  Human eats Candy  Enter command(product mode)  eat  Human do not eat Candy. Candy is spoiled  Enter command(product mode)  exit  Enter commmand  exit |

|  |
| --- |
| **Задача 2** |
| PS C:\Users\Yarik Scherbatyuk\Documents\OOP\_2course\_scherbatyukk\lab5\src\2> dotnet run  Enter amount  12345  Cash:  5x1  100x1  200x1  1000x12  20x2  Enter amount  12345,6  Unable to give cash, try enter another amount  Enter amount  exit |

**Обґрунтування використання шаблонів**

1. В даній задачі використано шаблон Стан, тому що поведінка об’єкта залежить від стану продукту: якщо період придатності продукту не пройшов, тоді його можна споживати, якщо пройшов – тоді не можна.
2. В даній задачі використано шаблон Ланцюжок обов’язків, тому що для видачі введеної суми необхідно, щоб обробник визначивши кількість купюр найбільшого номіналу, які потрібно видати, переходив до купюр меншого номіналу, визначав їх кількість та знову переходив до купюр меншого номіналу, тобто працював за принципом «оброби сам або передай іншому».

**Висновки**

В ході лабораторної роботи я покращив знання про поведінкові шаблони проектування. Навчився використовувати шаблони проектування в залежності від умов задачі.