

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 2**

з дисципліни “ Об’єктно-орієнтоване програмування ”

тема “Реалізація складних структурних шаблонів проектування.”

|  |  |  |
| --- | --- | --- |
| Виконала  студентка II курсу  групи КП-72  Городченко Анна Володимирівна  (*прізвище, ім’я, по батькові*) |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2019 р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Варіант №6

Київ 2019

**Мета роботи**

Ознаойомлення 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 завдання**

1. Розробити модуль до програмного забезпечення, яке

використовується в міліції. Даний модуль повинен забезпечити

автоматичне створення електронної справи затриманого (окрім загальної

анкети, де міститься загальна інформація про злочинця, дата та причини

його затримання) у разі, якщо його затримано більше, ніж на мінімальний

термін ув’язнення.

2. Система оплати за проїзд в метро. Пасажири можуть

сплачувати за допомогою жетонів, проїзних квитків на місяць та карток з

певною кількістю поїздок. Турнікет на вході дає можливість проходу

пасажиру, якщо він отримує дозвільний сигнал від системи контролю

проїзний документів. За допомогою шаблона проектування розробити

програмний модуль, який буде перетворювати отриманий платіж на сигнал

«1» («так») чи «0» («ні») для турнікету.

**Коротке обгрунтування вибору шаблону проектування**

***Завдання 1 ( Заступник ):***

Структурний шаблон «Заступник» я використала, оскільки він:

- зберігає посилання, яке дозволяє йому звертатися до реального суб'єкту, використовуючи інтерфейс класу *Offender*;

- додає щось до запиту перед його надсиланням до мережі або змінює запит;

- контролює доступ до *Offender*  через перехоплення всіх викликів

***Завдання 2 (Адаптер):***

Структурний шаблон «Адаптер» я використала, оскільки :

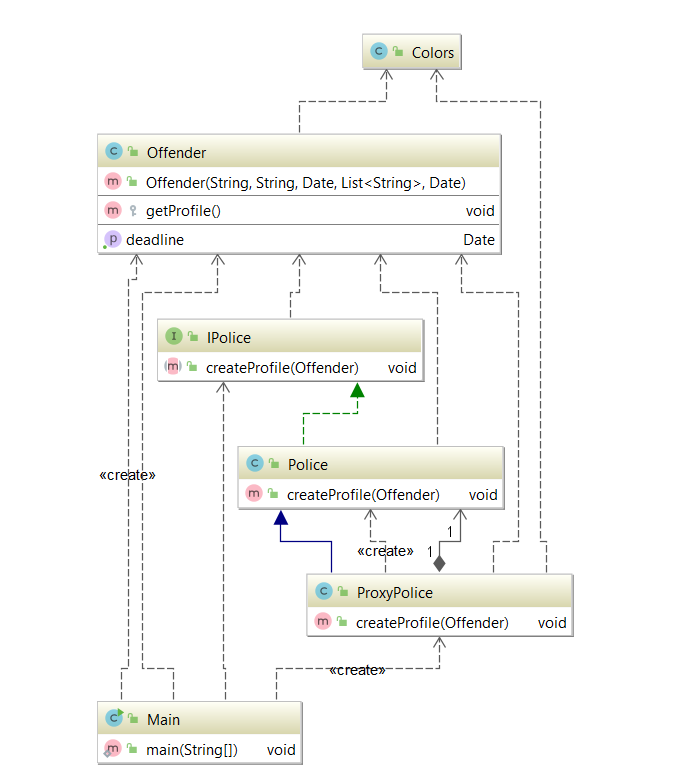
- існуючий клас підтримує необхідні дані і поведінку, але має інтерфейс, який не відповідає певним вимогам

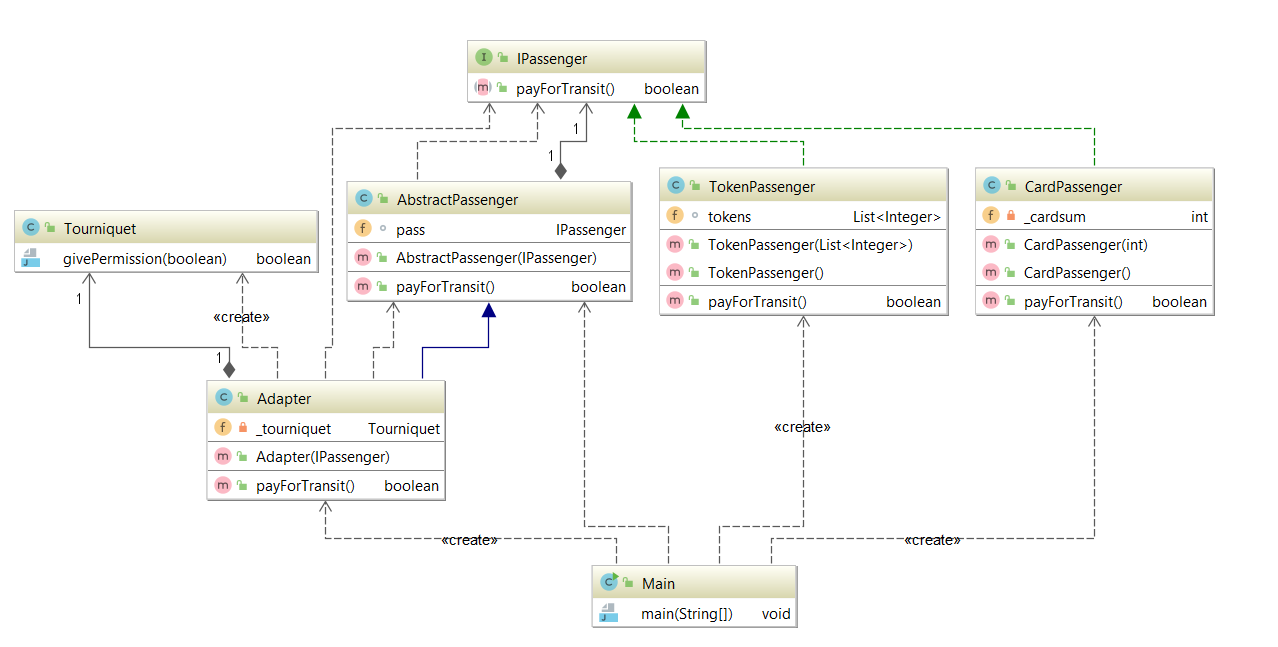
- необхідно створити повторно використовуваний клас, який повинен взаємодіяти із зазделегідь невідомими або непов’язаними з ним класами, які мають несумісні інтерфейси;

- необхідно об’єднати інтерфейси декількох класів, причому в наявності можуть бути тільки об’єкти-підкласи кожного з цих класів, а не примірники цих класів.

**Діаграма класів**

Завдання 1



Завдання 2

**Код класів**

Завдання 1

|  |
| --- |
| ProxyPolice.java |
| **package** task1;  **import** java.util.Date;  **public class** ProxyPolice **extends** Police {   **private** Police **police** = **new** Police();   @Override  **public void** createProfile(Offender offender) {   **if** (offender.getDeadline().before(**new** Date())) {   **police**.createProfile(offender);  System.***out***.println(Colors.***ANSI\_RED*** + **"Offender's deadline has already come, get him out!"** + Colors.***ANSI\_RESET***);   } **else** {  **police**.createProfile(offender);  }  } } |

|  |
| --- |
| Offender.java |
| **package** task1;   **import** java.text.SimpleDateFormat; **import** java.util.Date; **import** java.util.List;  **public class** Offender {  **private** String **firstName**;  **private** String **lastName**;  **private** Date **time\_of\_crime**;  **private** List<String> **reasons**;  **private** Date **deadline**;      **public** Offender(String firstName, String lastName, Date time\_of\_crime, List<String> reasons, Date deadline) {  **this**.**firstName** = firstName;  **this**.**lastName** = lastName;  **this**.**time\_of\_crime** = time\_of\_crime;  **this**.**reasons** = reasons;  **this**.**deadline** = deadline;  }   **public** Date getDeadline() {  **return deadline**;  }   **protected void** getProfile() {  SimpleDateFormat format = **new** SimpleDateFormat(**"yyyy-MM-dd"**);   System.***out***.println( Colors.***ANSI\_YELLOW*** + **" Offender profile"** + Colors.***ANSI\_RESET***);  System.***out***.println(**"Last name : "** + **lastName**);  System.***out***.println(**"First name : "** + **firstName**);  System.***out***.print(**"Reasons to be detained : "**);  **for** (String reason : **reasons**) {  System.***out***.print(**" "** + reason);  }  System.***out***.println(**"\nDate of crime : "** + format.format(**time\_of\_crime**));  System.***out***.println(**"Date of freedom : "** + format.format(**deadline**));   } } |
|  |
|  |

|  |
| --- |
| Main.java |
| **package** task1;  **import** java.text.ParseException; **import** java.text.SimpleDateFormat; **import** java.util.ArrayList; **import** java.util.Date; **import** java.util.List;  **public class** Main {  **public static void** main(String[] args) **throws** ParseException {   String pattern = **"yyyy-MM-dd"**;  SimpleDateFormat simpleDateFormat = **new** SimpleDateFormat(pattern);   Date time\_of\_crime = simpleDateFormat.parse(**"2018-09-09"**);  Date deadline = simpleDateFormat.parse(**"2019-01-09"**);    List<String> reasons = **new** ArrayList<>(2);  reasons.add(**"stealing"**);  reasons.add(**"attempted murder"**);   Offender offender = **new** Offender(**"Gatsby"**, **"Big"**, time\_of\_crime, reasons, deadline);   IPolice police = **new** ProxyPolice();  police.createProfile(offender);   } } |

Завдання 2

|  |
| --- |
| Adapter.java |
| **package** task2;  **public class** Adapter **extends** AbstractPassenger {  **private** Tourniquet **\_tourniquet** = **new** Tourniquet();   **public** Adapter(IPassenger p) {  **super**(p);  }   @Override  **public boolean** payForTransit() {  **boolean** resultOfPayment = **super**.payForTransit();  **return \_tourniquet**.givePermission(resultOfPayment);   } } |

|  |
| --- |
| Tourniquet.java |
| **package** task2;  **import** task1.Colors;  **public class** Tourniquet {   **public boolean** givePermission(**boolean** payment) {  **if** (payment) {  System.***out***.println(**"Signal from tourniquet : "** + Colors.***ANSI\_GREEN*** + **"Allowed"** + Colors.***ANSI\_RESET***);  **return true**;  } **else** {  System.***out***.println(**"Signal from tourniquet : "** + Colors.***ANSI\_RED*** + **"Forbidden"** + Colors.***ANSI\_RESET***);  **return false**;  }   } } |

|  |
| --- |
| AbstractPassenger.java |
| **package** task2;  **public class** AbstractPassenger {  IPassenger **pass**;   **public** AbstractPassenger(IPassenger p) {  **this**.**pass** = p;  }   **public boolean** payForTransit() {  **return this**.**pass**.payForTransit();  } } |

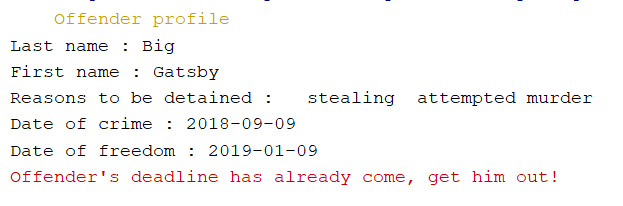
|  |
| --- |
| Main.java |
| **package** task2;  **import** java.util.ArrayList;  **public class** Main {  **public static void** main(String[] args) {   AbstractPassenger cardPassenger = **new** Adapter(**new** CardPassenger(5));  cardPassenger.payForTransit();   AbstractPassenger cardPassenger2 = **new** Adapter(**new** CardPassenger(400));  cardPassenger2.payForTransit();   AbstractPassenger tokenPassenger = **new** Adapter(**new** TokenPassenger(**new** ArrayList<>(0)));  tokenPassenger.payForTransit();   AbstractPassenger tokenPassenger2 = **new** Adapter(**new** TokenPassenger());  tokenPassenger2.payForTransit();    } } |

|  |
| --- |
| TokenPassenger.java |
| **package** task2;  **import** java.util.ArrayList; **import** java.util.Arrays; **import** java.util.List;  **public class** TokenPassenger **implements** IPassenger {  List<Integer> **tokens** ;   **public** TokenPassenger(List<Integer> tokens) {  **this**.**tokens** = tokens;  }   **public** TokenPassenger() {  **this**.**tokens** = **new** ArrayList<>(Arrays.*asList*(8,8,8));  }   @Override  **public boolean** payForTransit()  {  **if** (**tokens**.size() > 0)  {  **tokens**.remove(0);  **return true**;  }  **return false**;  } } |

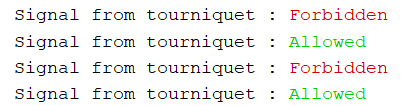
|  |
| --- |
| CardPassenger.java |
| **package** task2;  **public class** CardPassenger **implements** IPassenger {  **private int \_cardsum**;   **public** CardPassenger(**int** cardsum) {  **this**.**\_cardsum** = cardsum;  }   **public** CardPassenger() {  **this**.**\_cardsum** = 100;  }   @Override  **public boolean** payForTransit() {  **if** (**\_cardsum** >= 8) {  **\_cardsum** -= 8;  **return true**;  }  **return false**;  }  } |

**Приклади результатів**

Завдання 1:



Завдання 2:



**Висновки**

Виконавши дану лабораторну роботу, я ознайомилась з основними характеристиками шаблонів “Заступник”, “Фасад”, “Адаптер” та “Легковаговик”. Читання методичного посібника допомогло запам’ятовуванню поширених ситуацій, коли використання цих шаблонів є доцільним, виконання роботи сприяло набуттю вмінь та навичок реалізації шаблонів під час створення програмного коду.

Вибір шаблонів не був важким — у завданні явно відчувалися відтінки, які прослідковувалися в усіх раніше відомих мені прикладах випадків, де було доцільно використовувати обрані шаблони “Заступник” і “Адаптер”.