

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 № 5**

з дисципліни “ Об'єктно-орієнтовне програмування ”

тема “Реалізація поведінкових шаблонів проектування”

|  |  |  |
| --- | --- | --- |
| Виконав  студентка II курсу  групи КП-72  Городченко Анна Володимирівна  (*прізвище, ім’я, по батькові*)  варіант № 6 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2019

**Мета роботи**

Ознайомлення з основними характеристиками шаблонів «Ланцюжок обов’язків», «Команда» та «Посередник», запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

**Постановка завдання**

1. У автоматі з приготування гарячих напоїв (чай, кава тощо) є три баки з водою. На початку роботи автомату для обробки запиту клієнту використовується вода з першого баку. Якщо для виконання замовлення води, яка залишилася у баку, недостатньо, відбувається перемикання автомату на наступний бак. У випадку, якщо не залишилося води навіть у третьому баку, на екран автомата виводиться повідомлення про неможливість виконання замовлення клієнта. За допомогою шаблону проектування реалізувати роботу такого автомату.

2. Під час наукових досліджень зірку по небу ведуть декілька обсерваторій (адже у певний проміжок доби її краще видно у якомусь одному місці планети). В залежності від часу Центр спостереження за зірками передає контроль над зіркою від однієї обсерваторії до іншої. За допомогою шаблону проектування реалізувати механізм спостереження за зіркою, якщо відомо, що у Центра спостереження є тільки одна команда «Почати спостереження» і можливість змінювати виконавців даної команди.

**Обгрунтування вибору шаблонів проектування**

**1. Завдання 1**

a. у розроблюваній системі є група об’єктів, які можуть оброблювати запит певного типу, причому справжній обробник запиту заздалегідь невідомий;

b. потрібно відправити запит одному з декількох об’єктів, не вказуючи явно, якому саме;

c. всі повідомлення повинні бути оброблені хоча б одним об’єктом системи;

d. повідомлення в системі оброблюються за схемою “оброби сам або передай іншому”, тобто одні повідомлення обробляються на тому ж рівні, де вони були отримані, а інші – передаються об’єктам іншого рівня.

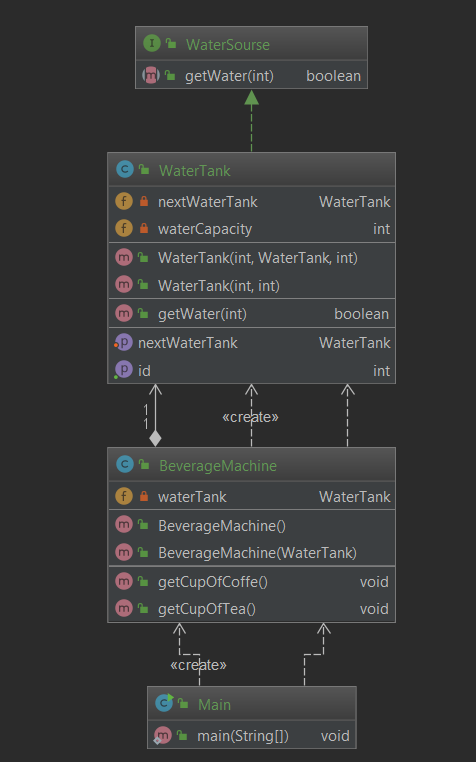
**2. Завдання 2**

a. команди для різних отримувачів повинні бути оброблені у різний спосіб.

b. потрібно визначати, ставити в чергу та виконувати запити у різний час;

Завдання 1

Діграма класів



**Текст коду**

|  |
| --- |
| package Chain |
| public class BeverageMachine {  private WaterTank waterTank;   public BeverageMachine() {  this.waterTank = new WaterTank(1, new WaterTank(2, 100), 50);  /\* or :  this.waterTank = new WaterTank(1, 50);  this.waterTank.setNextWaterTank(new WaterTank(2, 100));  \*/  }   public BeverageMachine(WaterTank waterTank) {  this.waterTank = waterTank;  }   public void getCupOfCoffe() {  if (waterTank.getWater(50)) {  System.*out*.println("Please, get your cup of coffe. Have a nice day!\n");  } else System.*out*.println("Technical problems, call the master!\n");  }   public void getCupOfTea() {  if (waterTank.getWater(100)) {  System.*out*.println("Please, get your cup of tea. Have a nice day!");  } else System.*out*.println("Technical problems, call the master!");  } } |
| public interface WaterSourse {  boolean getWater(int volume); } |
| public class WaterTank implements WaterSourse {  private int id;  private WaterTank nextWaterTank;  private int waterCapacity;   public int getId() {  return id;  }   public WaterTank(int id, WaterTank nextWaterTank, int capacity) {  this.id = id;  this.nextWaterTank = nextWaterTank;  this.waterCapacity = capacity;  }   public WaterTank(int id, int capacity) {  this.id = id;  this.waterCapacity = capacity;  }   public void setNextWaterTank(WaterTank nextWaterTank) {  if (this.nextWaterTank == null) this.nextWaterTank = nextWaterTank;  else System.*out*.println("The next water tank is set. It's id: #" + nextWaterTank.getId());  }   @Override  public boolean getWater(int volume) {   if (waterCapacity <= 0 || waterCapacity < volume) {  if (nextWaterTank == null) {  System.*out*.println("The water is over.");  return false;  } else {  return nextWaterTank.getWater(volume);  }  }  waterCapacity -= volume;  System.*out*.println("Getting water from tank #" + id);  return true;   } } |
| public class Main {  public static void main(String[] args) {  BeverageMachine machine = new BeverageMachine();// default capacity of water: 150   machine.getCupOfCoffe(); // 50  machine.getCupOfCoffe(); // 50  machine.getCupOfCoffe(); // 50  machine.getCupOfCoffe(); // 50   } } |

**Приклади виконання програми**

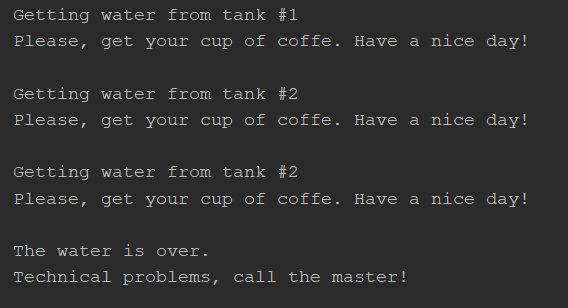
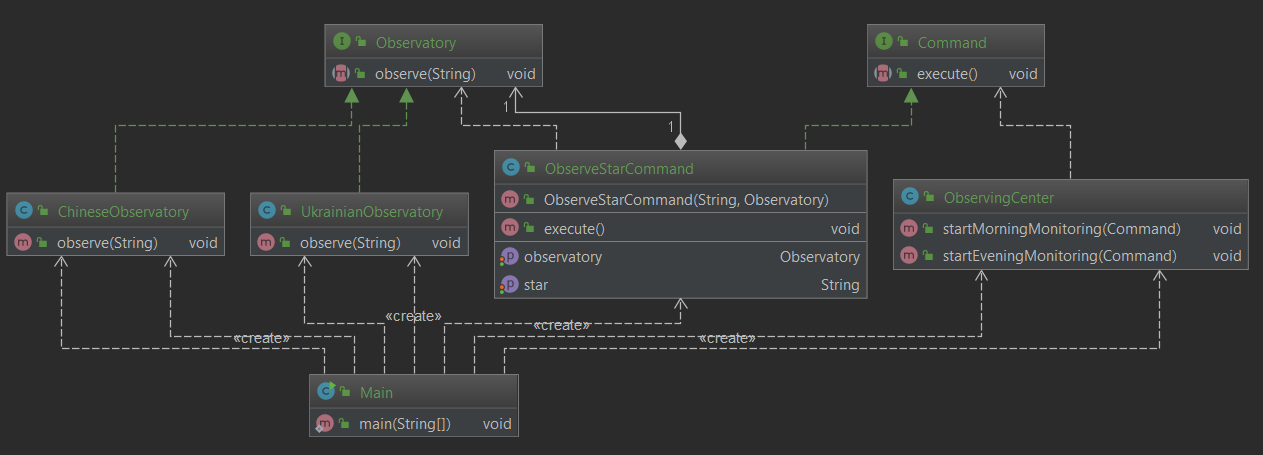


Рис. 1. Результат виконання програми

Завдання 2

Діаграма класів

**Текст програми**

|  |
| --- |
| package Command |
| public class ObserveStarCommand implements Command {  private String star;  private Observatory observatory;   public ObserveStarCommand(String star, Observatory observatory) {  this.star = star;  this.observatory = observatory;  }   public String getStar() {  return star;  }   public void setStar(String star) {  this.star = star;  }   public Observatory getObservatory() {  return observatory;  }   public void setObservatory(Observatory observatory) {  this.observatory = observatory;  }   @Override  public void execute() {  if (this.observatory != null) this.observatory.observe(this.star);  } } |
| public class UkrainianObservatory implements Observatory{  @Override  public void observe(String star) {  System.*out*.println("Ukrainian Observatory is observing " + star.toUpperCase());  } } |
| public interface Command {  void execute(); } |
| public class Main {  public static void main(String[] args) {  ObservingCenter observingCenter = new ObservingCenter();   ChineseObservatory chineseObservatory = new ChineseObservatory();  UkrainianObservatory ukrainianObservatory = new UkrainianObservatory();   System.*out*.println("Morning: ");  observingCenter.startMorningMonitoring( new ObserveStarCommand("Sirius", ukrainianObservatory));   observingCenter.startEveningMonitoring( new ObserveStarCommand("Sirius", chineseObservatory));  Timer.*setTimeout*(()-> System.*out*.println("Afternoon: "), 3000);    } } |

**Результати виконання програми**

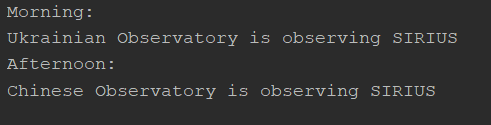


Рис. 2. Результат виконання програми

**Висновки**

Виконавши дану лабораторну роботу, я ознайомилась з основними

поведiнковими шаблонами «Ланцюжок обов’язків», «Команда» та

«Посередник». Я запам’ятала поширені ситуаціі, коли використання

цих шаблонів є доцільним. Виконання роботи сприяло набуттю вмінь та

навичок реалізації шаблонів під час створення програмного коду.