++rr

**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 орієнтоване програмування”**

**тема “Реалізація складних поведінкових шаблонів проектування”**

|  |  |  |
| --- | --- | --- |
| **Виконав**  **студент II курсу**  **групи КП-83**  **Коваль Андрій Олекснадрович**  **(прізвище, ім’я, по батькові)** |  | **Перевірив**  **“\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.**  **викладач**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **(прізвище, ім’я, по батькові)** |

**Київ 2020**

**Мета роботи**

Ознаойомлення 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**

Варіант 8

1. При покупці коштовної речі у магазині покупець виявив, що не має достатньої суми грошей і він повинен вирішити чи позичити гроші, щоб заплатити за покупку одразу, чи брати її у кредит. Він телефонує своєму другу з проханням позичити суму, якої не вистачає. У випадку, якщо друг має необхідну суму і може її позичити, покупець оформлює покупку одразу. Якщо друг не має потрібної суми, він може в свою чергу зателефонувати комусь зі своїх друзів з аналогічним проханням. У випадку, якщо суми не знайдено, покупець оформлює покупку у кредит. За допомогою шаблону проектування змоделювати описану ситуацію.

2. Для продажу/купівлі квартири люди звертаються до агентств нерухомості, де у випадку продажу квартиру реєструють у базі квартир та допомагають знайти на неї покупця, у випадку ж купівлі – реєструють у базі покупців, фіксують побажання покупця до квартири та підбирають множину квартир, які відповідають даним вимогам. За допомогою шаблона проектування змоделювати процес продажу/купівлі квартир за участі безпосередніх продавців та покупців, а також агентства нерухомості. При підборі пропозицій квартир агентство нерухомості може звертатися по додаткову інформацію до інших подібних агентств.

**Задача 1**

**Обґрунтування вибору шаблону**

Шаблон «Ланцюг відповідальності» використовується для передачі запитів послідовно через ланцюжок обробників. У нашому випадку запит йде від покупця до друзів. Якщо вони не допомагають, то йде далі до інших можливих друзів друзів, а в кінці, якщо ніхто не зміг допомогти, то береться кредит.

**Діаграма класів**

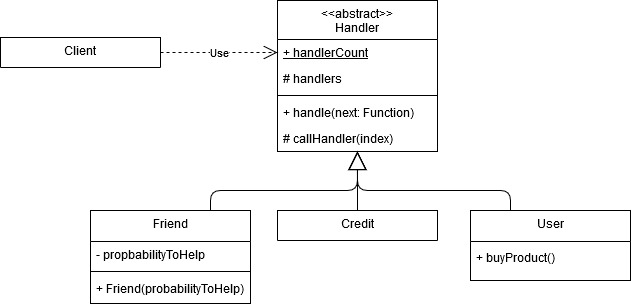
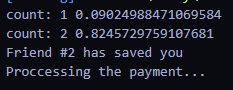


Рис. Діаграма класів завдання 1

**Фрагменти коду програми**

|  |
| --- |
| **Абстрактний клас обробника запиту** |
| **export** **abstract** **class** Handler {  **public** **static** handlerCount = **0**;  **protected** handlers: **Handler**[] = [];  **abstract** handle(next: **Function**);  **public** use(handler: **Handler**) {  **this**.handlers.push(handler, ...handler.handlers);  }  **protected** callHandler(index: **number**) {  **const** currHandler = **this**.handlers[index];  **if** (!currHandler) {  **return**;  }  currHandler.handle(() => **this**.callHandler(index + **1**));  }  } |

|  |
| --- |
| **Конкретны обробники** |
| **export** **class** Friend **extends** Handler {  **constructor**(**private** propabilityToHelp = Math.random()){  **super**();  }  **public** handle(next: **Function**) {  **const** count = ++Handler.handlerCount;  console.log('count:', count, **this**.propabilityToHelp);  **if** (Math.random() <= **this**.propabilityToHelp) {  console.log(`Friend #${count} has saved you`);  console.log('Proccessing the payment...');  } **else** {  **if** (!next != **null**) {  next();  } **else** {  console.log('Sorry, you cant buy anything without a credit');  }  }  }  }  **export** **class** Credit **extends** Handler {  **constructor**() {  **super**();  }  **public** handle(next: **Function**) {  console.log('A credit handler will supply you with all money you need');  console.log('Proccessing the payment...');  **if** (next)  next();  }  }  **export** **class** User **extends** Handler{  **constructor**() {  **super**();  }  **public** buyProduct() {  **this**.callHandler(**0**);  }  **public** handle(\_next: **Function**) {  **this**.buyProduct();  }  } |
| **Клієнтський код** |
| **const** user = **new** User();  **const** friend1 = **new** Friend();  **const** friend2 = **new** Friend();  user.use(**new** Friend());  friend1.use(friend2);  friend1.use(**new** Friend());  user.use(friend1);  user.use(**new** Friend());  user.use(**new** Credit());  user.buyProduct(); |

**Результати роботи програми**

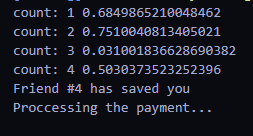


Рис. Вивід програми при декількох випадкових сценаріях

**Задача 2**

**Обґрунтування вибору шаблону**

Шаблон «Команда» використовується для перетворення запитів у об’єкти для більш простої подальшої обробки їх у обробниках. У нашому випадку запити виконують роль пошуку необхідних клієнтів на нерухомості спрощуючи код класу агенції з нерухомості

**Діаграма класів**

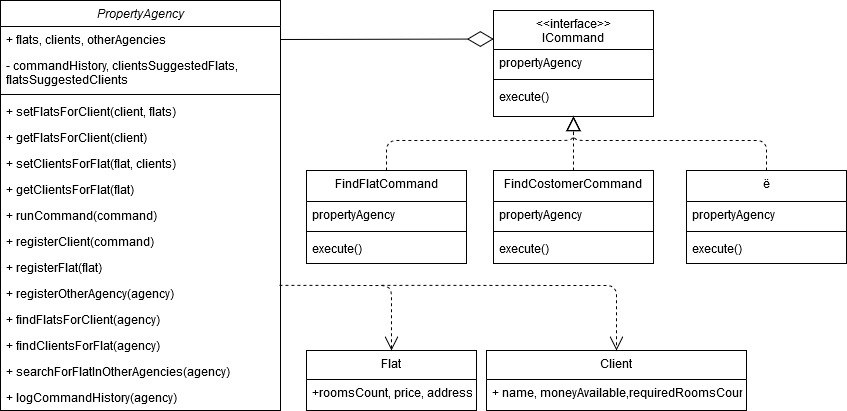


Рис. Діаграма класів завдання 2

**Фрагменти коду програми**

|  |
| --- |
| **Агентство з нерухомості** |
| **export** **class** PropertyAgency {  // ...  **private** runCommand(command: **ICommand**) {  command.execute();  **this**.commandHistory.push(command);  }  **public** registerClient(client: **Client**) {  console.log('Registrating new client', client.name);  **this**.\_clients.push(client);  }  **public** registerFlat(flat: **Flat**) {  **this**.\_flats.push(flat);  }  **public** registerOtherAgency(agency: **PropertyAgency**) {  **this**.\_otherAgencies.push(agency);  }  **public** findFlatsForClient(client: **Client**) {  **if** (!**this**.\_clients.includes(client))  **this**.registerClient(client);    **this**.runCommand(**new** FindFlatCommand(**this**, client));  console.log(client.name, **this**.getFlatsForClient(client));  }  **public** findClientsForFlat(flat: **Flat**) {  **if** (!**this**.\_flats.includes(flat)) {  console.log('No flat found, registrating one');  **this**.registerFlat(flat);  }    **this**.runCommand(**new** FindCostomerCommand(**this**, flat));  console.log(flat.address, **this**.getClientsForFlat(flat));  }  **public** searchForFlatInOtherAgencies(client: **Client**) {  **this**.runCommand(**new** FindFlatInOtherAgenciesCommand(**this**, client));  }  **public** logCommandHistory() {  console.log(**this**.commandHistory);  }  //...  } |

|  |
| --- |
| **ICommand** |
| **export** **interface** ICommand {  propertyAgency: **PropertyAgency**;  execute();  } |

|  |
| --- |
| **Різні види команд** |
| **export** **class** FindFlatCommand **implements** ICommand {  **constructor**(**public** propertyAgency: **PropertyAgency**, **private** client: **Client**) { }  execute() {  **const** flats = **this**.propertyAgency.flats.filter(  f => f.price <= **this**.client.moneyAvailable && f.roomsCount >= **this**.client.requiredRoomsCount  );  **this**.propertyAgency.setFlatsForClient(**this**.client, flats);  }  }  **export** **class** FindCostomerCommand **implements** ICommand {  **constructor**(**public** propertyAgency: **PropertyAgency**, **private** flat: **Flat**) {}  execute() {  **const** clients = **this**.propertyAgency.clients.filter(  c => c.moneyAvailable >= **this**.flat.price && c.requiredRoomsCount <= **this**.flat.roomsCount  );  **this**.propertyAgency.setClientsForFlat(**this**.flat, clients);  }  }  **export** **class** FindFlatInOtherAgenciesCommand **implements** ICommand {  **constructor**(**public** propertyAgency: **PropertyAgency**, **private** client: **Client**) { }  execute() {  **const** flats: **Flat**[] = [];  **this**.propertyAgency.otherAgencies.forEach(  ag => {  ag.findFlatsForClient(**this**.client);  flats.push(...ag.getFlatsForClient(**this**.client));  }  );  **this**.propertyAgency.setFlatsForClient(**this**.client, flats);  }  } |

|  |
| --- |
| **Клієнтський код** |
| **const** agency1 = **new** PropertyAgency();  **const** agency2 = **new** PropertyAgency();  clients.forEach(c => agency1.registerClient(c));  flats.slice(**0**, **2**) .forEach(f => agency1.registerFlat(f));  flats.slice(**2**).forEach(f => agency2.registerFlat(f));  agency1.registerOtherAgency(agency2);  agency1.findFlatsForClient(clients[**0**]);  agency1.findFlatsForClient(clients[**2**]);  agency1.searchForFlatInOtherAgencies(clients[**0**])  agency1.searchForFlatInOtherAgencies(clients[**2**])  flats.forEach(f => agency1.findClientsForFlat(f));  flats.forEach(f => agency2.findClientsForFlat(f));  // agency1.logCommandHistory(); |

**Результати роботи програми**

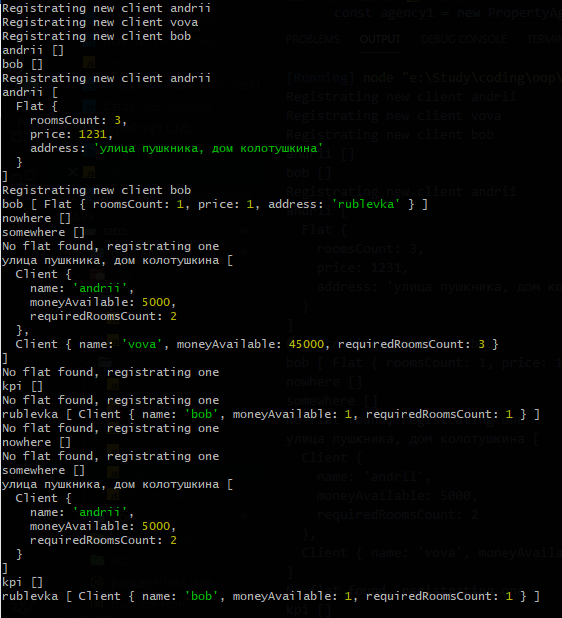


Рис. Приклад роботи програми

**Висновки**

Я ознайомився з основними характеристиками шаблонів «Ланцюжок обов’язків», «Команда» та «Посередник», запам’ятав поширені ситуації, коли використання цих шаблонів є доцільним, набув вмінь та навичок реалізації шаблонів під час створення програмного коду.