Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра автоматизованих систем управління



Методичні вказівки

до лабораторної роботи № 4

**«Моделювання класів»**

з дисципліни

**«Основи автоматизованого проектування складних об’єктів та систем»**

для студентів

базового напрямку підготовки бакалавр

за спеціальністю “Комп’ютерні науки” (шифр 6.050101)

Львів-2015

**Методичні вказівки до лабораторної роботи № 4 «Моделювання класів»** з дисципліни *“*Основи автоматизованого проектування складних об’єктів та систем *”* для студентів спеціальності - шифр **6.050101** *“Комп’ютерні науки”* Укл. Дорошенко А.В., Львів: Національний університ**ет “Львівська політехніка”, 2015**.

Методичні вказівки обговорено та схвалено на засіданні кафедри АСУ

Протокол № 11-2014/2015 від «27» березня 2015 р.

Методичні вказівки обговорено та схвалено на засіданні методичної комісії базового напрямку підготовки

Протокол № 8-14/15 від «23» квітня 2015 р.

**Лабораторна робота № 4**

**«Моделювання класів»**

***Мета роботи:*** Освоїти моделювання класів в UML-діаграмах та їх побудову у конкретному середовищі.

***Завдання:***

1. ***Оволодіти навичками моделювання класів для UML-діаграм***
2. ***Розробити діаграми класів для індивідуального завдання.***

**1. Теоретичні відомості**

На **діаграмах класів** показані різноманітні класи, які утворюють систему, а також їх взаємозв’язки. Діаграми класів називають “статичними діаграмами”, оскільки на них показано класи разом з атрибутами і операціями (методами), а також статичний взаємозв’язок між ними: те, яким класам «відомо» про існування яких класів, і те, які класи «є частиною» інших класів, — але не показано методи, які при цьому викликаються.

Діаграми класів дозволяють:

1. На концептуальному рівні формувати «словник предметної області»;
2. На рівні специфікації і реалізації визначати структуру класів у програмній реалізації системи.

Діаграми класів можуть використовуватись для генерації каркасного програмного коду ( в реальній мові програмування).

**Клас** визначає атрибути і методи набору об’єктів. Всі об’єкти цього класу (екземпляри цього класу) мають спільну поведінку і однаковий набір атрибутів (кожен з об’єктів має свій власний набір значень).

У UML класи позначаються прямокутниками з назвою класу, у цих прямокутниках у вигляді двох “відсіків” може бути показано атрибути і операції (методи) класу:

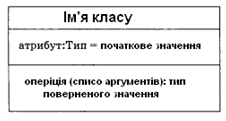


Рис. 1 Наочне представлення класу у UML

**Атрибути** показуються щонайменше назвою. Також може бути показано їх тип, початкове значення і інші властивості. У синтаксисі мови UML це виглядає таким чином: <*видимість> <ім'я>: <тип>* = <*значення за умовчанням>.*

Області видимості атрибута:

* + відповідає *публічним (public)* атрибутам;
* # відповідає *захищеним (protected)* атрибутам;
* - відповідає *приватним (private)* атрибутам.

**Операціями (методами)** є процеси, що реалізовуються деяким класом. Вони також показуються принаймні назвою. Крім того, може бути показано їх параметри і типи значень, які буде повернуто. Повний синтаксис операцій в мові UML виглядає таким чином:

<*видимість> <ім'я> (<список параметрів>): <вираз, який повертає значення типу> {<рядок властивостей>}.*

Області видимості операції:

* + відповідає *публічним (public)* операціям;
* # відповідає *захищеним (protected)* операціям;
* - відповідає *приватним (private)* операціям.

Список параметрів містить розділені комою параметри, синтаксис яких аналогічний синтаксису атрибутів: <напрям*> <ім'я>: <тип> = Значення за умовчанням>.* При цьому додатковим елементом є *напрям,* який застосовується, щоб показати характер використання параметра - для входу (in), виходу *(out)* або в обох напрямах *(inout).* Якщо значення *напряму* відсутнє, воно передбачається вхідним (in).

Вираз, який повертає значення типу містить список розділених комою значень типів.

Рядок властивостейвказує значення властивостей, які застосовуються до даної операції.

**Способи зв’язку класів**

Класи можна співвіднести (пов’язати) один з одним у декілька способів:

1. **Узагальнення.**

Однією з фундаментальних основ об’єктно-орієнтованого програмування є наслідування. Нащадок отримує всі атрибути і операції батьківського класу і може перевизначити або змінювати деякі з них, а також додавати власні атрибути і операції.

У UML пов’язування *Узагальнення* між двома класами розташовує їх у вузлах ієрархії, яка відповідає концепції успадкування класу-нащадка від базового класу. У UML узагальнення буде показано у вигляді лінії, яка поєднує два класи, зі стрілкою, яку спрямовано від базового класу.



Рис. 2 Наочне представлення узагальнення у UML.

1. **Асоціації.**

Асоціація означає взаємозв’язок між класами, вона є базовим семантичним елементом і структурою для багатьох типів “з’єднань” між об’єктами.

Асоціації є тим механізмом, який надає об’єктам змогу обмінюватися даними між собою. Асоціація описує з’єднання між різними класами (з’єднання між дійсними об’єктами називається об’єктним з’єднанням, або *зв’язком*).

Асоціації можуть виконувати роль, яка визначає призначення асоціації і може бути одно- чи двосторонньою (другий варіант означає, що у межах зв’язку кожен з об’єктів може надсилати повідомлення іншому, перший же — варіанту, коли лише один з об’єктів знає про існування іншого). Крім того, кожен з кінців асоціації має значення численності, яке визначає кількість об’єктів на відповідному кінці асоціації, які можуть мати зв’язок з одним з об’єктів на іншому кінці асоціації.

У UML асоціації позначаються лініями, що з’єднують класи, які беруть участь у зв’язку, крім того, може бути показано роль і численність кожного з учасників зв’язку. Численність буде показано у вигляді діапазону [мін..макс] невід’ємних чисел, зірочка (\*) на боці максимального значення позначає нескінченність.



Рис. 3 Наочне представлення асоціації у UML.

1. **Агрегації.**

Агрегації є особливим типом асоціацій, за якого два класи, які беруть участь у зв’язку не є рівнозначними, вони мають зв’язок типу “ціле-частина”. За допомогою агрегації можна описати, яким чином клас, який грає роль цілого, складається з інших класів, які грають роль частин. У агрегаціях клас, який грає роль цілого, завжди має численність рівну одиниці.

У UML агрегації буде показано асоціаціями, у яких з боку цілої частини буде намальовано ромб.



Рис. 4 Наочне представлення агрегації у UML.

1. **Композиції.**

Композиції — це асоціації, які відповідають *дуже сильній* агрегації. Це означає, що у композиціях ми також маємо справу з співвідношеннями ціле-частина, але тут зв’язок є настільки сильним, що частини не можуть існувати без цілого. Вони існують лише у межах цілого, після знищення цілого буде знищено і його частини.

У UML композиції буде показано як асоціації з зафарбованим ромбом з боку цілого.



Рис. 5 Наочне представлення композиції у UML.

**Приклади діаграм класів**

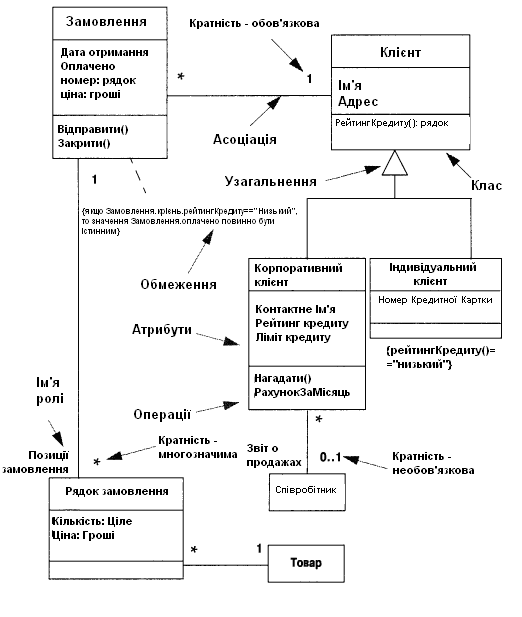


Рис.6. Діаграма класів

1. **Атрибути.** Наприклад, для класу Замовлення атрибутами є Дата отримання, Оплачено, Номер, Ціна. Кожен з цих атрибутів має свій тип.
2. **Операції.** Наприклад, для класу Замовлення операціями є Відправити() і Закрити (). Ці операції є без параметрів.
3. **Асоціації.** На цій діаграмі асоціація відбувається між класами Замовлення і Клієнт; Корпоративний клієнт і Співробітник; Замовлення і Рядок замовлення; Рядок замовлення і товар. Замовлення може поступити тільки від одного Клієнта. А Клієнт протягом деякого часу може зробити декілька Замовлень. Кожне з цих Замовлень може містити декілька Рядків замовлення, причому кожен Рядок замовлення повинен відповідати єдиному Товару. Символ «*\*»* біля класу Замовлення для асоціації між класами Замовлення і Клієнт показує, що з одним клієнтом може бути зв'язане багато замовлень; навпаки, символ «1» показує, що кожне із замовлень може поступити тільки від одного клієнта.
4. **Узагальнення.** Класи Корпоративний клієнт і Індивідуальний клієнт є нащадками (підкласами) класу Клієнт. Вони наслідують атрибути і операції батьківського класу, а також додають власні.

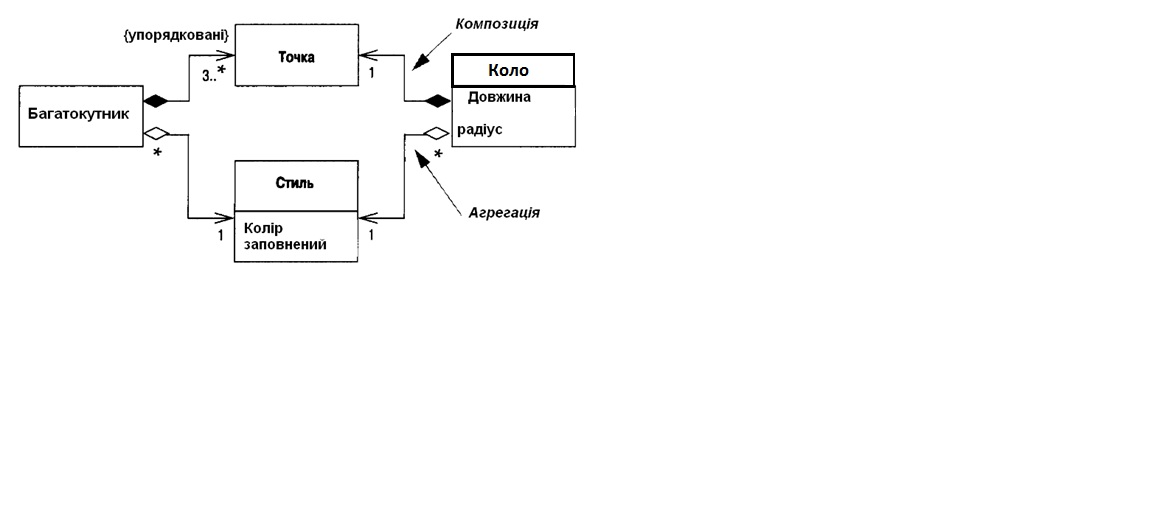


Рис. 7 Агрегація і композиція

На рис. 7 зображено приклад агрегації і композиції.

1. **Композиція.** Будь-який екземпляр Точки може бути або Багатокутником, або Колом, але не може бути ними одночасно. Вказана композиція означає, що видалення деякого Багатокутника спричинить видалення всіх асоційованих з ним Точок.
2. **Агрегація.** Деякий екземпляр Стилю може бути загальним для декількох Багатокутників і Кіл. В цьому випадку, якщо видалити деякий Багатокутник, то це не спричинить видалення асоційованого з ним Стилю, оскільки Стиль може існувати без Багатокутника.

**Інші елементи діаграми класів**

Окрім класів на діаграмах класів можуть міститися і деякі інші елементи.

* **Інтерфейси**

Інтерфейси — це абстрактні класи, тобто з них не можна напряму створювати екземпляри. У інтерфейсах можуть міститися операції, але не атрибути. Класи можуть бути нащадками інтерфейсів (за допомогою асоціації реалізації), а з цих діаграм можна потім створювати сутності.

* **Типи даних**

Типи даних — це базові елементи, з яких типово будується мова програмування. Типовими прикладами є цілі числа і булеві значення. Вони не можуть мати зв’язків з класами, але класи можуть мати зв’язки з ними.

* **Переліки**

Переліки є простими списками значень. Типовим прикладом є перелік днів тижня. Пункти переліків називаються літералами переліків. Подібно до типів даних, переліки не можуть мати зв’язків з класами, але класи можуть мати зв’язки з переліками.

* **Пакунки**

Пакункам відповідають простори назв у мовах програмування. На діаграмі пакунки використовуються для позначення частин системи, у яких міститься декілька класів, може навіть сотні класів.

Таблиця 2.1 Відповідність функціональних специфікацій і класів «Готель»

|  |  |  |
| --- | --- | --- |
| № | Специфікації | Клас |
| 1.  2.  3.  4. | Дає можливість клієнту провести пошук номерів, які б задовільняли його вимоги, за типом, вартістю номера. На пошуковій формі відображаються поля для заповнення, а також кнопка «Search».  Дає можливість на реєстрацію та бронювання в готелі, здійснити оплату та обрати перелік послуг(як стандартних так і додаткових), а також зберігати дані про оплату, обрані послуги та ін.  Можливість оновлення(редагування) бази даних номерів готелю, а також в класі зберігаються дані про кожен номер в готелі, його статус і дані про всіх клієнтів, які проживають в номерах.  Можливість звернення із системними повідомленнями(запитами) до бази даних і до клієнта, а також слідкування за наданням клієнту всіх, обраних ним, послуг. | Пошук (Пошук по базі даних)  Клієнт (Реєстрація, бронювання, оплата, послуги)  База даних  Сервіс(Системні повідомлення, обслуговування) |

**3. Порядок виконання роботи**

1. Ознайомитися з теоретичною частиною.
2. Ознайомитися із середовищем розробки діаграм.
3. Розробити діаграму прецедентів для свого індивідуального завдання.
4. Здійснити документацію для кожного прецеденту діаграми.
5. Оформити звіт по результатах виконаної роботи.

Вимоги до звіту

Оформити звіт для захисту лабораторної роботи за зразком:

* назва роботи
* мета роботи
* порядок роботи
* короткі теоретичні відомості
* аналіз отриманих результатів та висновок.

Рекомендована література

1. Язык UML. Руководство пользователю / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. 2-е изд., Питер 2004.
2. Головатий О.О. Методичні вказівки до оформлення пояснювальних записок із дипломних робіт, літньої практики, курсових робіт та рефератів для студентів спеціальностей “Програмне забезпечення автоматизованих систем” та “Економічна кібернетика”. Жовті Води, ІП “Стратегія”, 2005.
3. Хабуш А., Ткачук Н.В., Исмаилов Р. Применение Web–технологии в  информационно-управляющей системе газокомпрессорной станции магистрального газопровода. // Вісник ХДПУ. Харків, ХДПУ, вип. 102, 2000- С.113-117.
4. 1.    Hauswirth M., Jazayeri M. Schneider M. A phase model for e-commerce business models and its application to security assessment. Technical University of Vienna, Information Systems Institute. TUV-1841-00-04 June 5, 2000.
5. 2.    Kobryn C. Modeling Components and Frameworks with UML. Communications of the ACM 43(10). 2000. p31-38.

Методичні вказівки до лабораторної роботи № 4 *“ Моделювання класів”* з дисципліни *“*Основи автоматизованого проектування складних об’єктів та систем *”* для студентів спеціальності 6.050101 *“Комп’ютерні науки”*

Укладач:

**Дорошенко А.В.**

Комп’ютерний набір, верстку та редагування

здійснили ст. гр. КН-410, каф. АСУ, Мастикаш О.В. , Тринкін М.Є.