

Практична робота 4.

Побудова статичної моделі з використанням UML діаграм

Мета роботи: Вивчити методи побудови моделі процесів у вигляді UML діаграм

Теоретичні положення.

UML (Скор. від англ. Unified Modeling Language - уніфікована мова моделювання) - мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення.

UML є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, званої UML моделлю.

UML був створений для визначення, візуалізації, проектування та документування в основному програмних систем.

Використання UML не обмежується моделюванням програмного забезпечення. Його також використовують для моделювання бізнес-процесів, системного проектування та відображення організаційних структур.

UML дозволяє розробникам ПО досягти угоди в графічних позначеннях для представлення загальних понять (таких як клас, компонент, узагальнення (generalization), об'єднання (aggregation) і поведінку) і більше сконцентруватися на проектуванні та архітектурі.

В рамках мови UML всі уявлення про моделі складної системи фіксуються у вигляді спеціальних графічних конструкцій, що одержали назву діаграм, які доповнюються «механізмами розширення».

Діаграма (diagram) - графічне представлення сукупності елементів моделі у формі зв'язного графа, вершинам і ребрам (дугам) якого приписується певна семантика.

Нотація UML являє собою графічну інтерпретацію семантики для її візуального представлення.

Діаграми

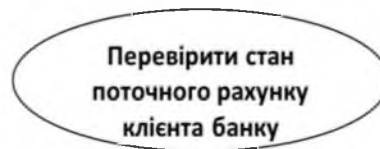
Діаграма	призначення	Тип діаграми (моделі IC)		
		за ступенем фізичної реалізації	По відображенню динаміки	по відображеному аспекту
Варіантів використання (Use case)	Відображає функції системи, взаємодія між акторами і функціями	логічна	динамічна	функціональна
Класів (Class)	Відображає набір класів, інтерфейсів і відносин між ними	логічна або фізична	статична	Функціонально-інформаційна

Діаграми Варіантів використання(Use Case).

Суть даної діаграми складається в наступному: система, що проектується представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) називається будь-яка сутність, що взаємодіє з системою з зовні. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на модельовану систему так, як визначить сам розробник.

У свою чергу, **варіант використання (use case)** служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який використовується системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою.

Окремий **варіант використання** позначається на діаграмі **еліпсом**, у середині якого міститься його коротка назва або ім'я у формі дієслова з пояснювальними словами.



Кожен варіант використання відповідає окремому сервісу, який надає модельовану сутність або систему по запиту користувача (актора), тобто визначає спосіб застосування цієї сутності. Сервіс, який здійснюється за запитом користувача, є закінченою послідовністю дій.

На діаграмі варіантів використання **інтерфейс** зображується у вигляді **маленького кола**, поруч з яким записується його ім'я. В якості імені може бути іменник, який характеризує відповідну інформацію або сервіс (наприклад, «форма замовлення», «інформація про клієнта»).



Суцільна лінія в цьому випадку вказує на той факт, що пов'язаний з інтерфейсом варіант використання повинен реалізовувати всі операції, необхідні для даного інтерфейсу, а можливо і більше (а). Крім цього, інтерфейси можуть об'єднуватися з варіантами використання пунктирною лінією зі стрілкою (б), яка означає, що варіант використання призначений для специфікації тільки того сервісу, який необхідний для реалізації даного інтерфейсу.

В мові UML примітки призначені для включення в модель довільної текстової інформації, що має безпосереднє відношення до контексту розроблюваного проекту.

Графічно примітки позначаються прямокутником з «загнутим» верхнім правим кутком. У середині прямокутника міститься текст примітки. Примітка може відноситися до будь-якого елементу діаграми, в цьому випадку їх з'єднує пунктирна лінія.

Якщо в примітці зазначається ключове слово «constraint», то дана примітка є обмеженням, що накладається на відповідний елемент моделі, але не на саму діаграму.



Між компонентами діаграми варіантів використання можуть існувати різні відносини, які описують взаємодію екземплярів одних акторів і варіантів використання з екземплярами інших акторів і варіантів. Один актор може взаємодіяти з декількома варіантами використання. В цьому випадку цей актор звертається до кількох сервісів даної системи. У свою чергу один варіант використання може взаємодіяти з декількома акторами, надаючи для всіх них свій сервіс.

У мові UML є кілька стандартних видів відносин між акторами і варіантами використання:

- **Відношення асоціації (association)**
- **Відношення розширення (extend)**
- **Відношення узагальнення (generalization)**
- **Відношення включення (include)**

Відношення Асоціації

Це відношення встановлює, яку конкретну **роль грає актор** при взаємодії з екземпляром варіанту використання. Відношення асоціації позначається суцільною лінією між актором і варіантом використання. Ця лінія може мати додаткові умовні позначення, такі, наприклад, як ім'я та кратність (**multiplicity**).



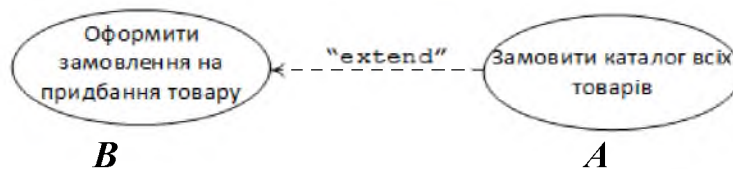
Форми запису кратності:

Ціле невід'ємне число (0999 ...)
Два цілих невід'ємних числа, розділені двома крапками (Гр1..Гр2)
Два символи, розділені двома крапками. При цьому перший з них є цілим невід'ємним числом або 0, а другий - спеціальним символом "*".
Єдиний символ "*", який є скороченням записи інтервалу «0 .. *»

Відношення Розширення

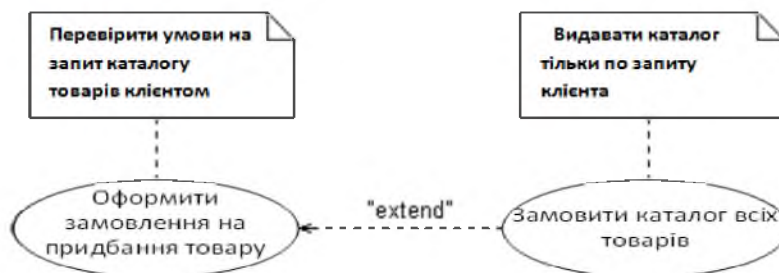
Визначає взаємозв'язок екземплярів окремого варіанту використання з більш загальним варіантом

Відношення розширення від варіанту використання **A** до варіанту використання **B** означає, що властивості екземпляра варіанту використання **B** можуть бути доповнені завдяки наявності властивостей у розширеного варіанту використання **A**, який *доповнює властивості варіанту B*.



Відношення розширення відзначає той факт, що один з варіантів використання може приєднувати до своєї поведінки деяку додаткову поведінку, певне для іншого варіанту використання.

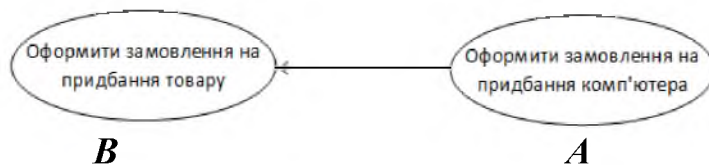
При оформленні замовлення на придбання товару тільки в деяких випадках може знадобитися надання клієнту каталогу всіх товарів. При цьому умовою розширення є запит від клієнта на отримання каталогу товарів.



Відношення Узагальнення

Відношення узагальнення служить для вказівки того факту, що деякий варіант використання **A** може бути узагальненим до варіанту використання **B**. В цьому випадку варіант **A** буде спеціалізацією варіанту **B**. При цьому **B**

називається предком або батьком по відношенню А, а варіант А - нащадком по відношенню до варіанту використання В.



Графічно дане відношення позначається суцільною лінією зі стрілкою в формі незафарбованого трикутника, яка вказує на батьківський варіант використання

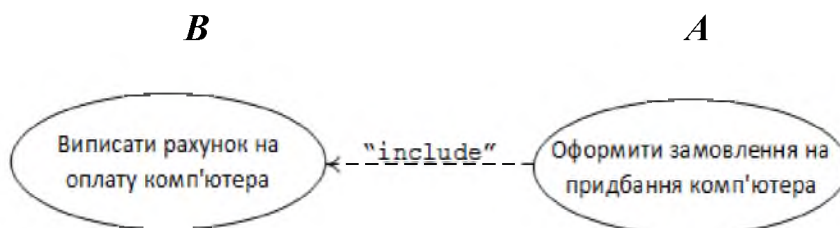
Між окремими акторами також може існувати відношення узагальнення.



Відношення Включення

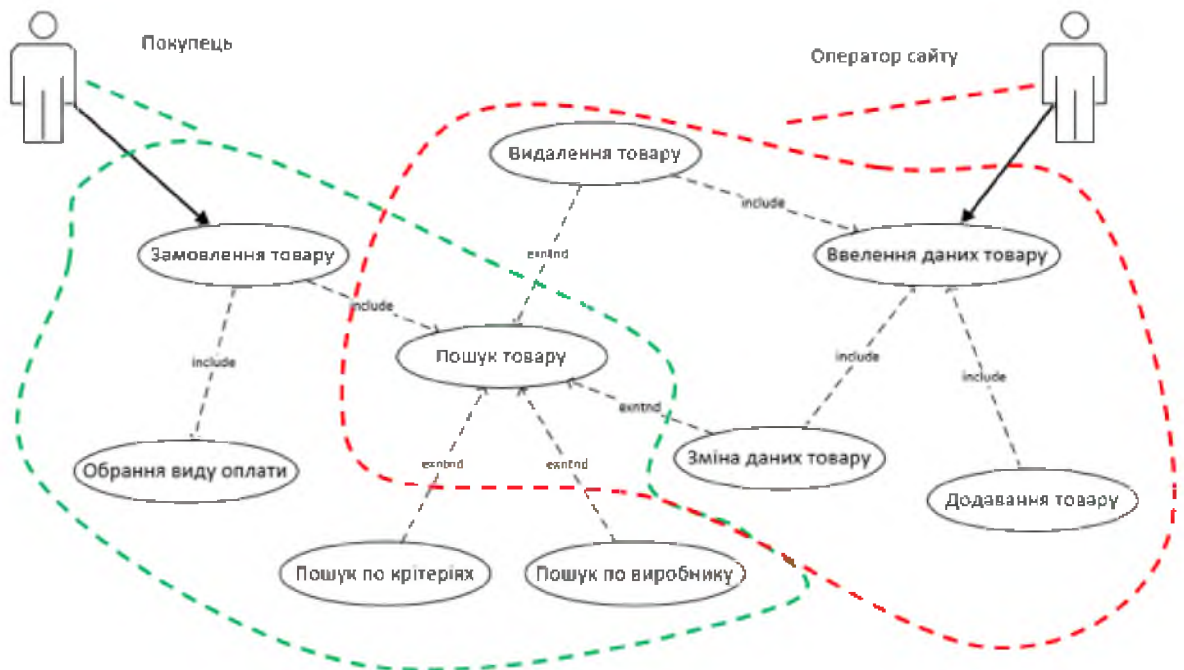
Відношення включення між двома варіантами використання вказує, що деяка задана поведінка для одного варіанта використання включається як складовий компонент в послідовність поведінки іншого варіанту використання. Відношення включення, спрямоване від варіанту використання А до варіанту використання В, вказує, що кожен екземпляр варіанту А включає в себе функціональні властивості, задані для варіанту В.

Графічно дане відношення позначається пунктирною лінією зі стрілкою (варіант відношення залежності), спрямованою від базового варіанту використання до такого, що включається. При цьому дана лінія зі стрілкою позначається ключовим словом «include» («включає»)



Зони відповідальності сценаріїв дій

Зони відповідальності сценаріїв дій визначаються виділенням на діаграмі варіантів використання груп елементів, які мають відношення до певного актора (ролі). Таких груп може бути декілька. Групи, що відносяться штриховою лінією певного кольору. До одного актора може мати відношення декілька груп з однаковим позначенням. Наприклад.




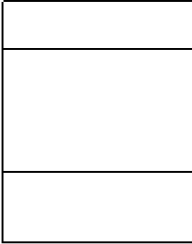

Діаграма класів (class diagram)

Діаграма класів (class diagram) служить для представлення *статичної структури моделі* системи в термінології класів об'єктно-орієнтованого програмування.

Діаграма класів може показувати різні взаємозв'язки між окремими сутностями предметної області, такими *як об'єкти і підсистеми*, а також описує їх *внутрішню структуру і типи відносин*. Діаграма класів є *графом*, вершинами якого є *елементи*, які *пов'язані* різними типами *структурних відносин*.

Діаграма класів може містити *інтерфейси, пакети, відносини* і навіть окремі екземпляри, такі як *об'єкти і зв'язки*

Сутності

Найменування	позначення	Визначення (семантика)
Клас (Class)		Безліч об'єктів, що мають загальну структуру і поведінку
об'єкт (Object)		Абстракція реальної чи уявної сутності з чітко вираженими концептуальними межами, індивідуальністю (ідентичністю), станом і поведінкою. З точки зору UML об'єкти є екземплярами класу (екземплярами сутності)
Інтерфейси (Interface)		Набір операцій, що визначає сервіс (набір послуг), що надається класом або компонентом

Клас (class) в мові UML служить для позначення об'єктів, які мають однакову структуру, поведінку і відносини з об'єктами інших класів. Графічно клас зображується у вигляді прямокутника, який додатково може бути розділений горизонтальними лініями на розділи або секції. У цих розділах можуть зазначатися *ім'я класу, атрибути (змінні) і операції (методи)*.

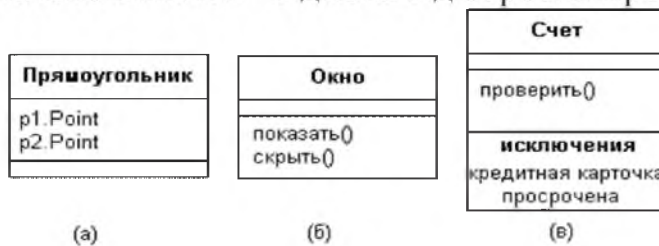
На початку розробки діаграми класи можуть позначатися прямокутником зі зазначенням тільки імені класу (а). А при уточненні та доробці компонентів діаграми опису класів у описі доповнюються атрибути (б) і операції (в).



У першому випадку для класу «Прямокутник» (а) вказані тільки атрибути - точки на координатній площині, які визначають його розташування.

Для класу «Вікно» (б) вказані тільки операції, секція атрибутів залишена порожньою.

Для класу «Рахунок» (в) додатково зображена четверта секція, в якій зазначено виняток - відмова від обробки простроченої кредитної картки.



Ім'я класу має бути *унікальним* одного опису, що описує сукупністю діаграм класів.

Воно вказується в першій верхній секції прямокутника. На додаток до загального правила найменування елементів мови UML, ім'я класу записується *по центру секції імені напівжирним шрифтом* і має *починатися з великої літери*.

Для позначення *імена класів* використовуються *іменники*.

Прикладами імен класів можуть бути такі іменники, як «Співробітник», «Компанія», «Керівник», «Клієнт», «Продавець», «Менеджер», «Офіс».

Коли необхідно явно вказати, до якого *пакету* належить той чи інший клас, то використовується спеціальний символ роздільник – подвійна двокрапка «::».

Синтаксис рядка імені класу в цьому випадку буде наступний *<Имя_Пакета> :: <ім'я_класу>*. Іншими словами, перед ім'ям класу має бути явно вказане ім'я пакету, до якого його слід віднести.

У *другій секції зверху* прямокутника класу записуються його *атрибути (attributes) або властивості*.

Кожному атрибуту класу відповідає *окремий рядок тексту*, яка складається з квантора видимості атрибута, імені атрибута, його кратності, типу значень атрибута і його вихідного значення (не завжди: *<Квантор видимості> <ім'я атрибута> [кратність] : <Тип атрибута> = <початкове значення> {рядок-властивість}*

Квантор видимості може приймати одне з трьох можливих значень.

Символ "+" позначає атрибут з областю видимості типу загальнодоступний (public). Атрибут з цією областю видимості доступний або видний з будь-якого іншого класу пакета, в якому визначена діаграма.

Символ "#" позначає атрибут з областю видимості типу захищений (protected). Атрибут із цією областю видимості недоступний або не видимий для всіх класів, за винятком підкласів даного класу.

Символ "-" позначає атрибут з областю видимості типу закритий (private). Атрибут із цією областю видимості недоступний або не видимий для всіх класів без виключення.

Квантор видимості можна не вказувати.

Замість умовних графічних позначень можна записувати відповідне ключове слово: **public, protected, private.**

Ім'я атрибута є рядком тексту, який є ідентифікатора відповідного атрибута і повинен бути *унікальним* в межах даного класу.

Ім'я атрибута є *єдиним обов'язковим елементом позначення атрибута.*

Кратність атрибута вказує *кількість атрибутів даного типу*, що входять до складу класу.

Кратність записується в формі рядка тексту в квадратних дужках після імені відповідного атрибута : [Нижня_границя1 .. верхня_границя1, нижня_границя2 .. верхня_границя2, ..., нижня_границя k .. верхня_границя k], де нижня границя і верхня границя є *позитивними цілими числами*.

В якості верхньої межі може використовуватися *спеціальний символ "*" , який означає довільне натуральне число.*

Якщо кратність атрибута не вказана, то за замовчуванням приймається її значення рівне 1..1

Початкове значення служить для завдання деякого *початкового значення* для відповідного атрибута в момент створення *окремого екземпляра класу.*

Рядок-властивість служить для вказівки значень атрибута, *які не можуть бути змінені* в програмі при роботі з даним типом об'єктів.

Фігурні дужки позначають фіксоване значення відповідного атрибута для класу в цілому, яке повинні приймати всі новостворювані екземпляри класу без винятку.

У третій зверху секції прямокутника записуються *операції або методи класу.*

Операція (operation) являє собою *сервіс*, що надає кожний екземпляр класу за певними умовами.

Кожній операції класу відповідає *окремий рядок*, який складається з *квантора видимості операції, імені операції, вирази типу значення, що повертається операцією*, та можливо, *рядок-властивість* даної операції :

<Квантор видимості> <ім'я операції> (список параметрів) : <Вираз типу значення, що повертається> {рядок-властивість}

Квантор видимості, як і в випадку атрибутів класу, може приймати одне з трьох можливих значень.

Символ "+" позначає операцію з областю видимості типу загальнодоступний (public).

Символ "#" позначає операцію з областю видимості типу захищений (protected).

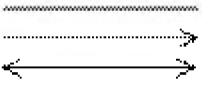
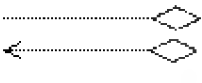

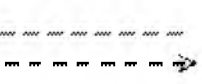

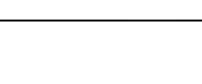
Символ "-" використовується для позначення операції з областю видимості типу закритий (private). Квантор видимості для операції може бути опущений.

Крім внутрішнього устрою або структури класів на відповідній діаграмі вказуються різні **відношення між класами**. При цьому сукупність типів таких відношень фіксована в мові UML і зумовлена семантикою цих типів відношень.

Базовими відношеннями або зв'язками в мові UML є:

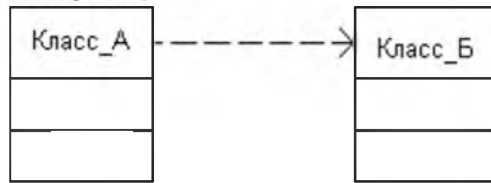
- **Відношення залежності (dependency relationship)**
- **Відношення асоціації (association relationship)**
- **Відношення узагальнення (generalization relationship)**
- **Відношення реалізації (realization relationship)**

Кожне з цих відношень має власне графічне представлення на діаграмі, яке відображає взаємозв'язки між об'єктами відповідних класів.

Найменування	позначення	Визначення (семантика)
Асоціація (association)		Відношення, яке описує значимий зв'язок між двома і більше сутностями. Найбільш загальний вигляд відносини
Агрегація (aggregation)		Підвид асоціації, яка описує зв'язок «частина» - «ціле», в якому «частина» може існувати окремо від «цілого». Ромб вказується з боку «цілого». Ставлення вказується тільки між сутностями одного типу
Композиція (composition)		Підвид агрегації, в якій «частини» не можуть існувати окремо від «цілого». Як правило, «частини» створюються і знищуються одночасно з «цілим»
Залежність (dependency)		Відношення між двома сутностями, в якому зміна в одній сутності (незалежної) може впливати на стан або поведінку іншої сутності (залежної). З боку стрілки вказується незалежна сутність
Узагальнення (generalization)		Відношення між узагальненою сутністю (предком, батьком) і спеціалізованою сутністю (нащадком, донькою). Трикутник вказується з боку батька. Ставлення вказується тільки між сутностями одного типу
Реалізація (realization)		Відношення між сутностями, де одна сутність визначає дію, яке інша сутність зобов'язується виконати. Відносини використовуються в двох випадках: між інтерфейсами і класами (або компонентами), між варіантами використання і кооперації. З боку стрілки вказується сутність, що б вплив (інтерфейс або варіант використання)

Відношення залежності використовується в такій ситуації, коли деяка **зміна одного елемента моделі** може зажадати **зміни іншого залежного від нього елемента моделі**.

Відношення залежності графічно зображується пунктирною лінією між відповідними елементами зі стрілкою на одному з її кінців («->» або «<-»). **На діаграмі класів дане відношення зв'язує окремі класи між собою**, при цьому стрілка спрямована **від класу-клієнта залежності до незалежного** класу або класу-джерела).



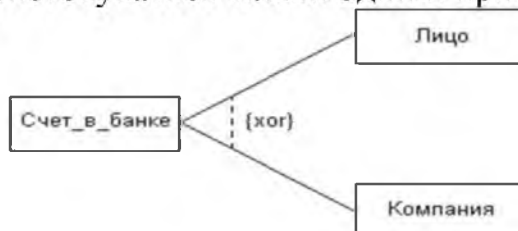
Відношення асоціації відповідає наявності деяких відносин між класами. Дане відношення позначається суцільною лінією з додатковими спеціальними символами, які характеризують окремі властивості конкретної асоціації.

Найбільш простий випадок даного відносини - бінарна асоціація. Вона пов'язує два класи.

Для бінарної асоціації на діаграмі може бути вказаний порядок проходження класів з використанням трикутника у формі стрілки поруч з ім'ям даної асоціації.

Напрямок стрілки вказує на **порядок класів**, один з яких є першим (з боку трикутника), а інший - другим (з боку вершини трикутника).

Окремим випадком відносини асоціації є так звана **виключна асоціація (Xor-association)**. Семантика даної асоціації вказує на той факт, що з кількох потенційно можливих варіантів даної асоціації в кожен момент часу може використовуватися тільки один її примірник.



Відношення агрегації має місце між **декількома класами** в тому випадку, якщо один з класів є деякою сутністю, що включає в себе **в якості складових частин інші сутності**. Дане відношення має фундаментальне значення для опису структури складних систем, оскільки застосовується для відображення **системних взаємозв'язків типу «частина-ціле»**.

Як приклад відносини агрегації розглянемо взаємозв'язок типу «частина-ціле», яка має місце між сутністю «Персональний комп'ютер» і такими компонентами, як «Системний блок», «Монітор», «Клавіатура», «Миш».

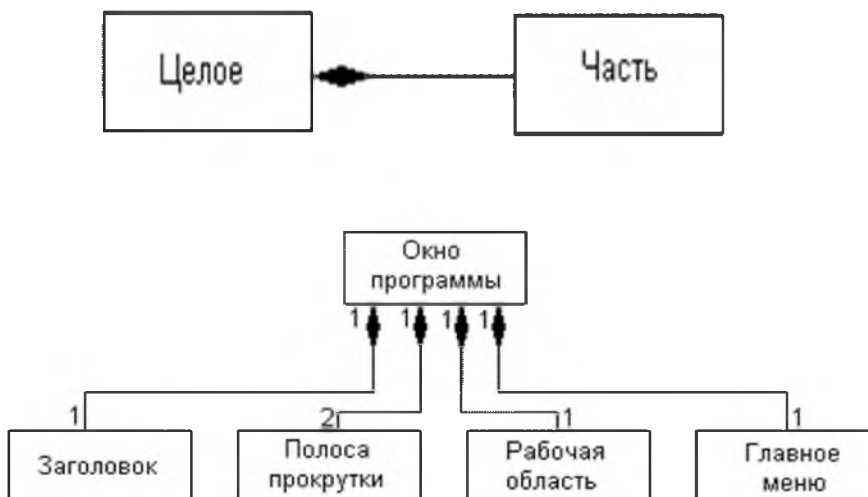
Графічно відношення агрегації зображується суцільною лінією, один з кінців якої являє собою зафарбований усередині ромб. Цей ромб вказує на той з класів, який являє собою «ціле». Решта класи є його «частинами».



Відношення композиції є окремим випадком **відносини агрегації**.

Це відношення служить для виділення **спеціальної форми відносини «частина-ціле»**, при якій складові частини в деякому сенсі перебувають **всередині цілого**.

Специфіка взаємозв'язку між ними полягає в тому, що **частини не можуть виступати у відриві від цілого**, тобто, зі знищенням цілого знищуються і всі його складові частини.



Відношення узагальнення є звичайним відношенням між **більш загальним елементом** (батьком або предком) і **більш приватним або спеціальним елементом** (дочірнім або нащадком).

Дане відношення може використовуватися для подання взаємозв'язків між пакетами, класами, варіантами використання і іншими елементами мови UML.

Дане відношення описує **ієрархічну будову класів** та успадкування їх **властивостей і поведінки**. При цьому передбачається, що клас-нащадок має всі властивості і поведінкою класу-предку, а також має свої власні властивості і поведінку, які відсутні у класу-предку.



Реалізація (realization) - це семантичне відношення між класифікаторами, при якому *один класифікатор визначає* зобов'язання, а інший гарантує його виконання.

Відношення реалізації зустрічаються в двох випадках:

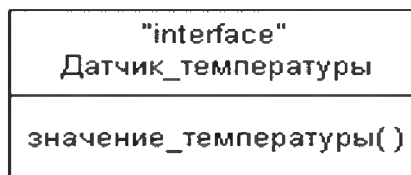
- по-перше - *між інтерфейсами* і реалізують їх класами або компонентами,
- а по-друге - *між прецедентами* і реалізують їх кооперації (конкретними зв'язками між конкретними класами - так наприклад "співпрацювати" можуть як дві компанії так і компанія і конкретна людина).

Відношення реалізації зображується у вигляді *пунктирною лінії з незафарбовані стрілкою (трикутником на кінці)*, як щось середнє між відносинами узагальнення і залежності - стрілка спрямована на "шаблон" - тобто на ту сутність, яка задає правила власної реалізації для сутності (реалізована), яка ці правила виконує (реалізує).

Інтерфейси є елементами діаграми варіантів використання.

Однак при побудові *діаграми класів окремі інтерфейси можуть уточнюватися* і в цьому випадку для їх зображення використовується спеціальний графічний символ - прямокутник класу з ключовим словом або стереотипом «interface».

При цьому секція атрибутів у прямокутника відсутня, а вказується лише секція операцій.



Об'єкт (object) є *окремим екземпляром класу*, який створюється на етапі виконання програми. Він має своє власне ім'я і конкретні значення атрибутів.

В силу різних причин може виникнути необхідність показати взаємозв'язку не тільки між класами моделі, а й *між окремими об'єктами, що реалізують ці класи*.

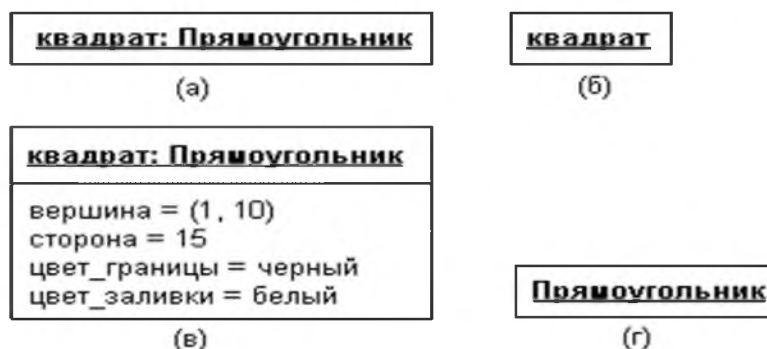
Для графічного зображення об'єктів використовується такий же символ прямокутника, що і для класів.

Відмінності проявляються при вказівці імен об'єктів, які в разі об'єктів обов'язково підкреслюються.

При цьому запис імені об'єкта є рядком тексту «ім'я об'єкта: ім'я класу», розділену двокрапкою (рис. А, б).

Ім'я об'єкта може бути відсутнім, в цьому випадку передбачається, що об'єкт є анонімним, і двокрапка вказує на дану обставину (рис. Г).

Відсутні може і ім'я класу. Тоді вказується просто ім'я об'єкта (рис. В). Атрибути об'єктів приймають конкретні значення.



Завдання: (за варіантами практичних робіт №1-3)

1. Побудуйте діаграму варіантів використання (Use case). Діаграма має включати не менш 10 сутностей 'Варіант використання', не менш 5 сутностей 'Акторів' та необхідні інтерфейси і примітки.

Діаграма має включати всі (чотири) типи відношень.

2. Виділіть на діаграмі варіантів використання зони відповідальності кожного з акторів.

3. Опишіть варіанти використання у вигляді таблиці.

№	Варіанти використання	Функції	Пов'язані варіанти використання	Пов'язані актори чи інтерфейси (коли є)
1				
...				

4. Опишіть акторів у вигляді таблиці.

№	Актори	Функції
1		1. 2.
...		

5. Побудуйте діаграму класів (*class diagram*). Діаграма має включати не менш 10 сутностей 'класів', 'об'єктів' та 'інтерфейсів'.

6. Діаграма має включати не менш чотирьох типів відношень.

7. Опишіть діаграму класів у вигляді таблиці.

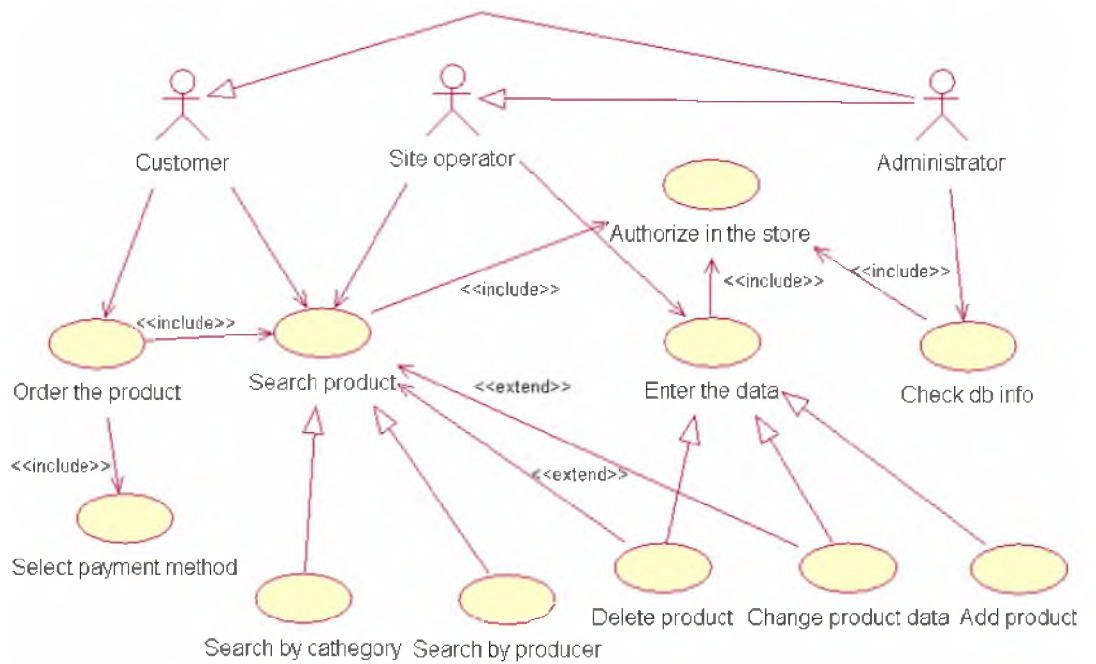
№	Клас	Пов'язані сутності	Тип відношення
1		1 2 ...	1 2 ...
...			

Приклади діаграм варіантів використання (Use case).

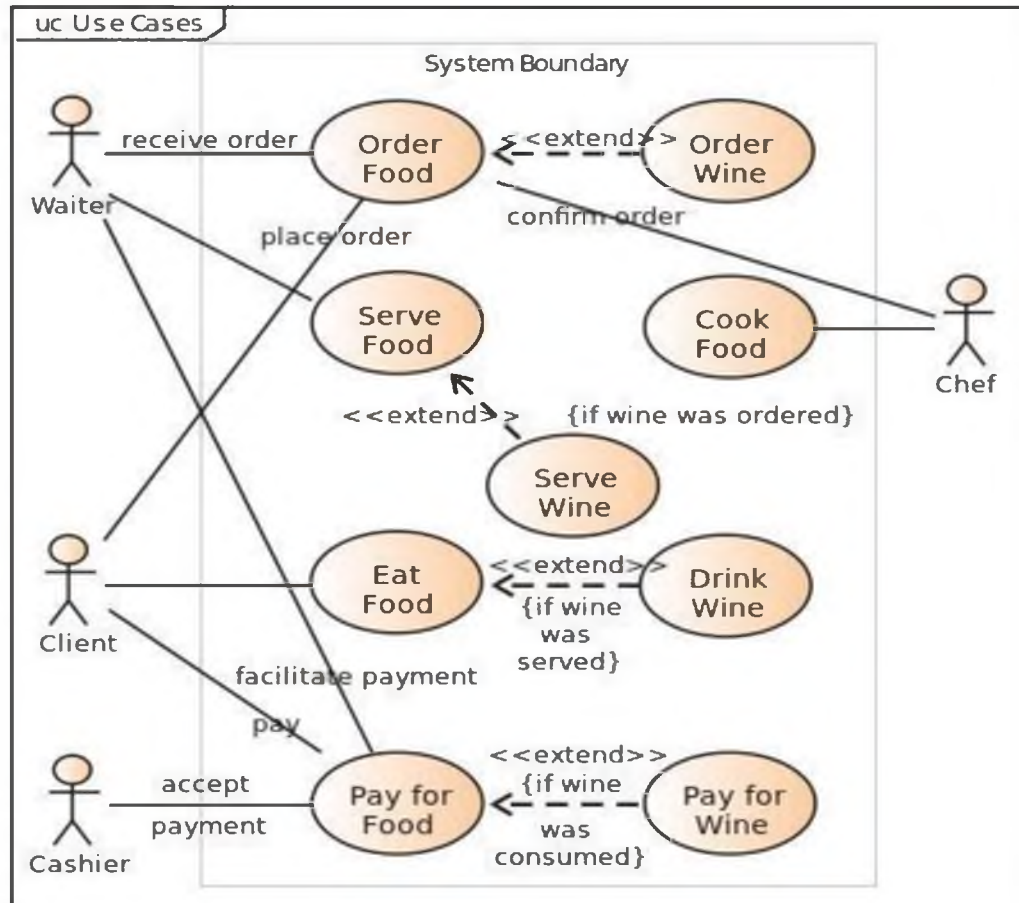
1. Пошук товару



2. Інтернет магазин.



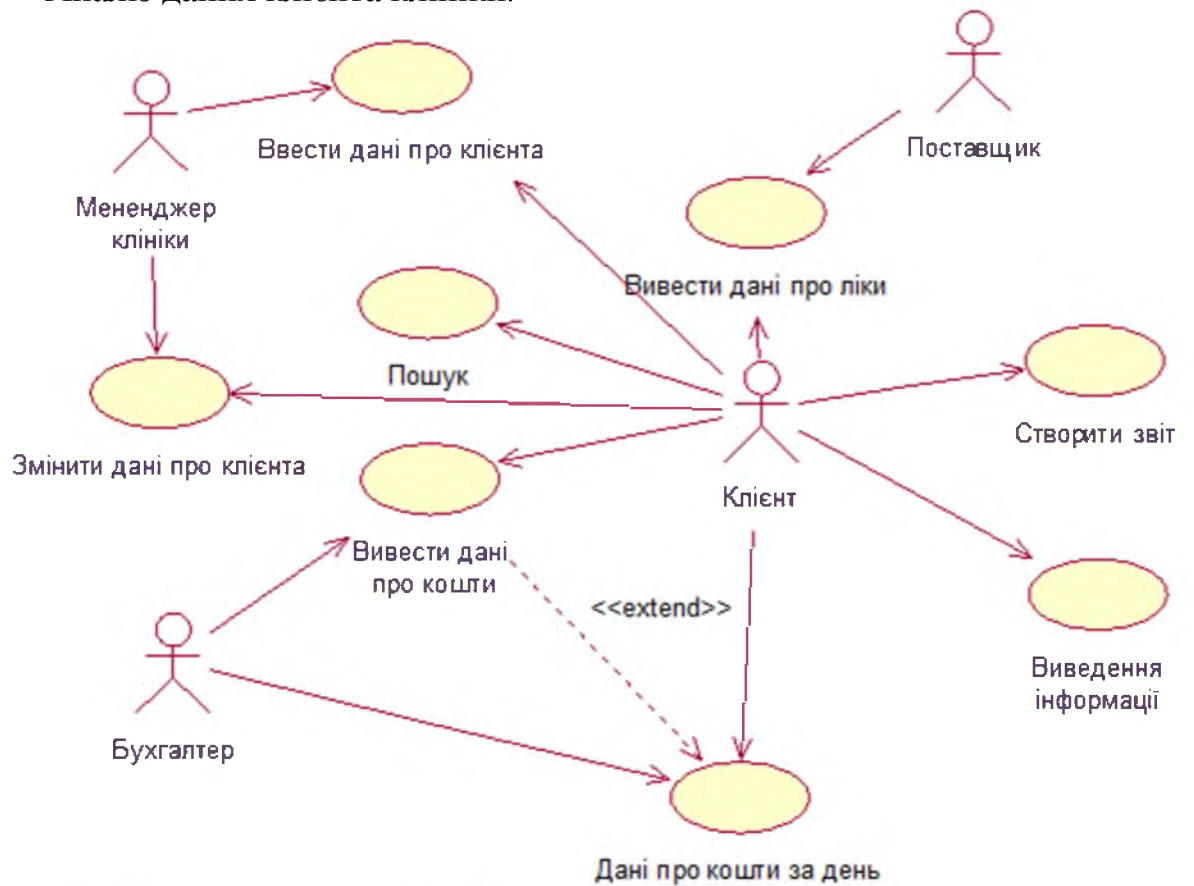
2. Обслуговування клієнта в ресторані



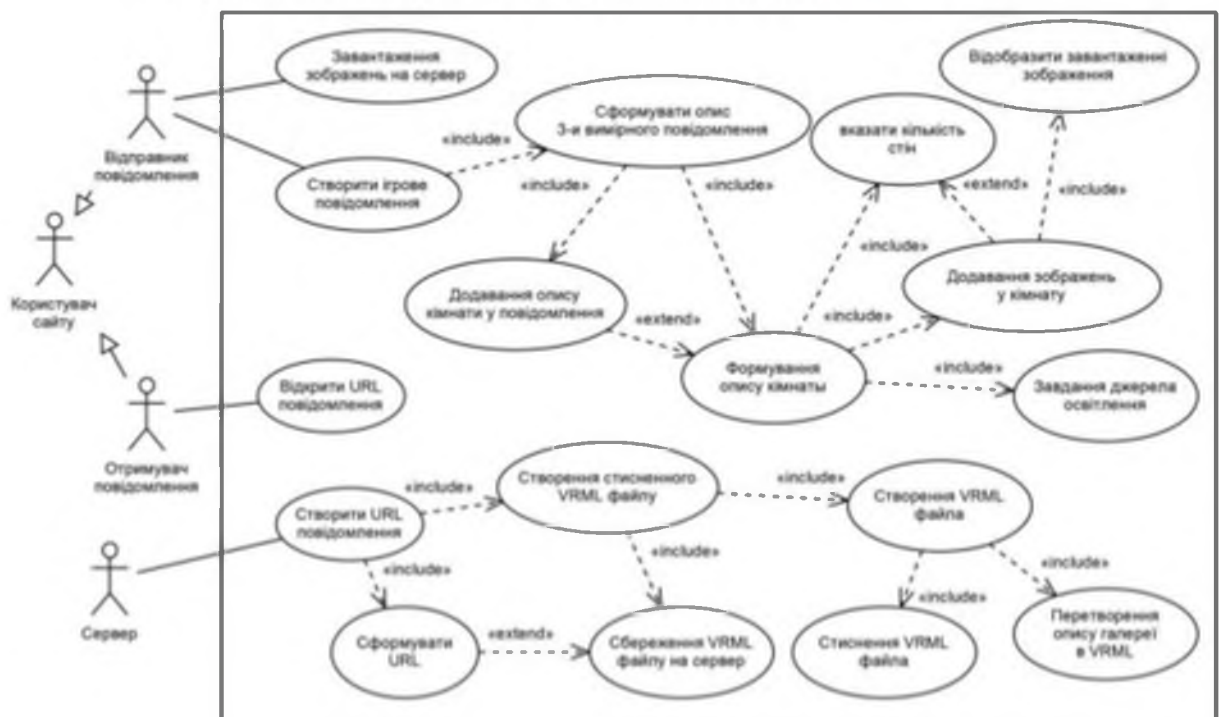
4. Продаж товару по каталогу



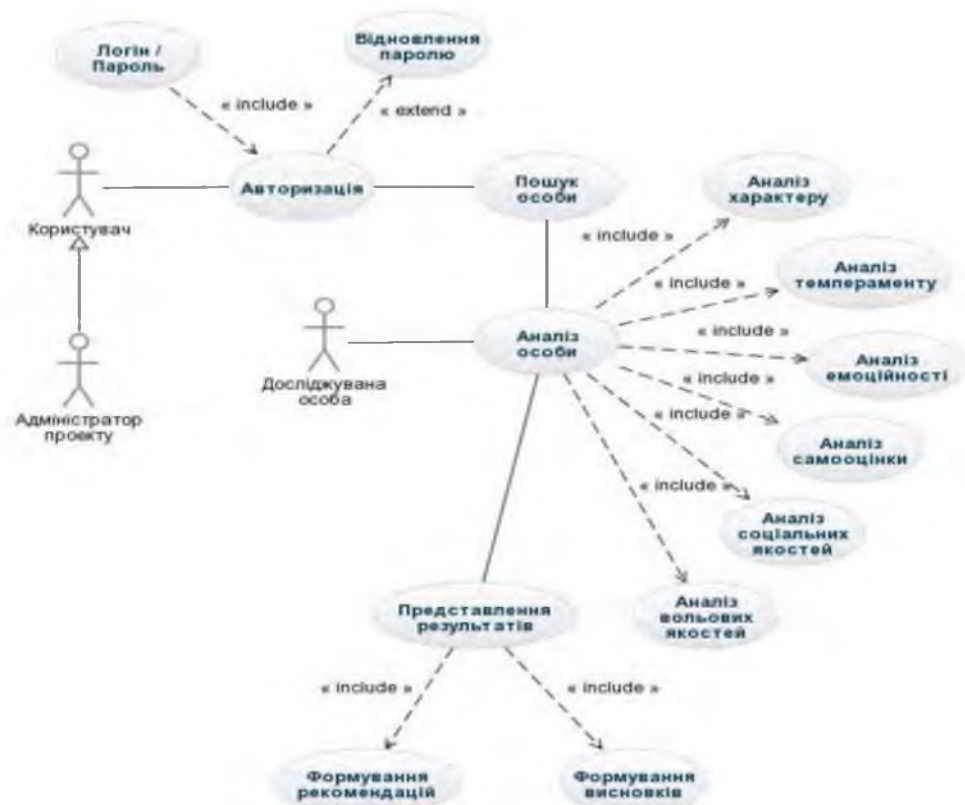
Аналіз даних клієнта клініки.



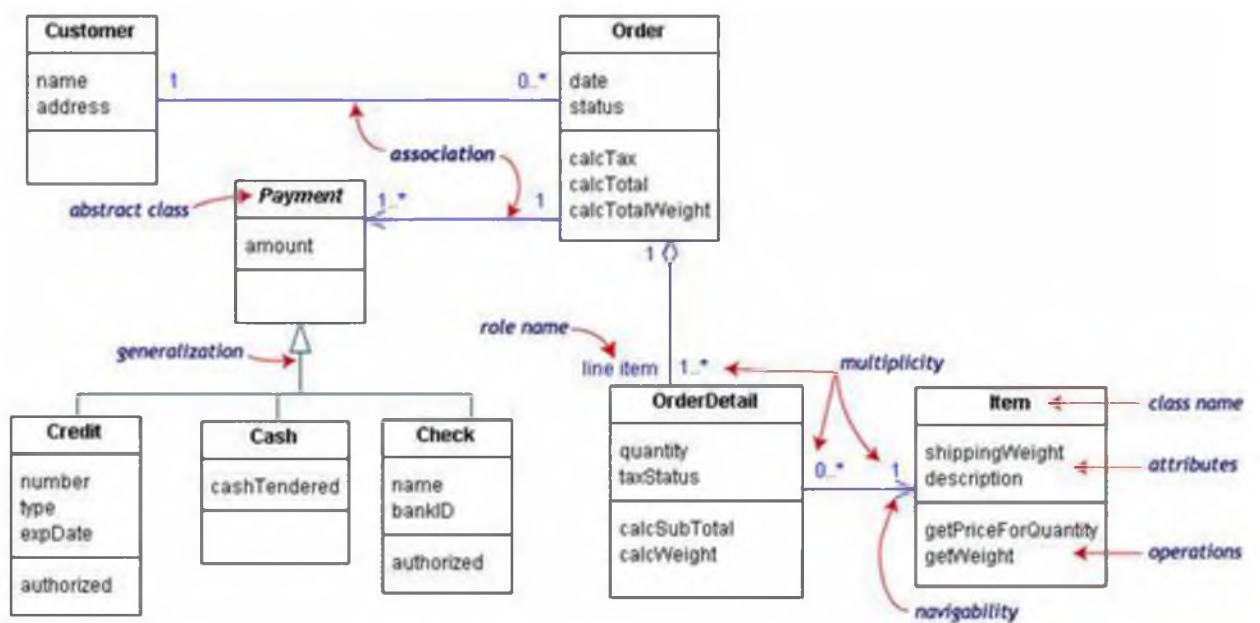
Розробка ПЗ створення віртуальних галерей



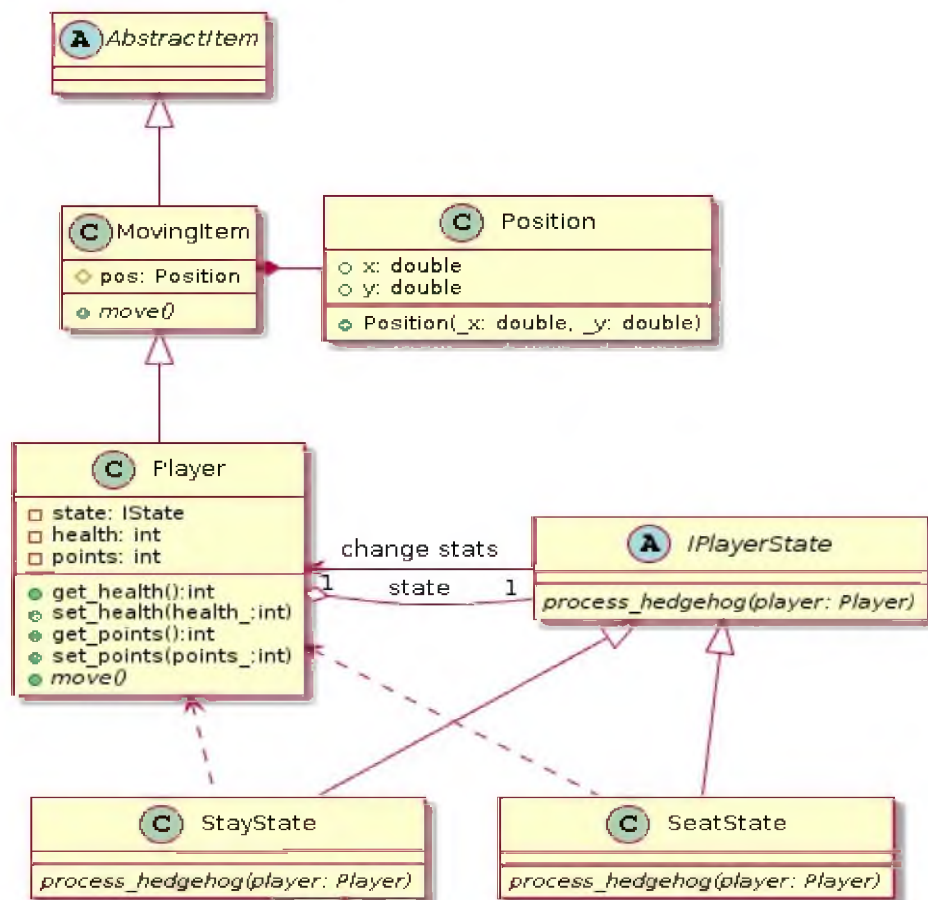
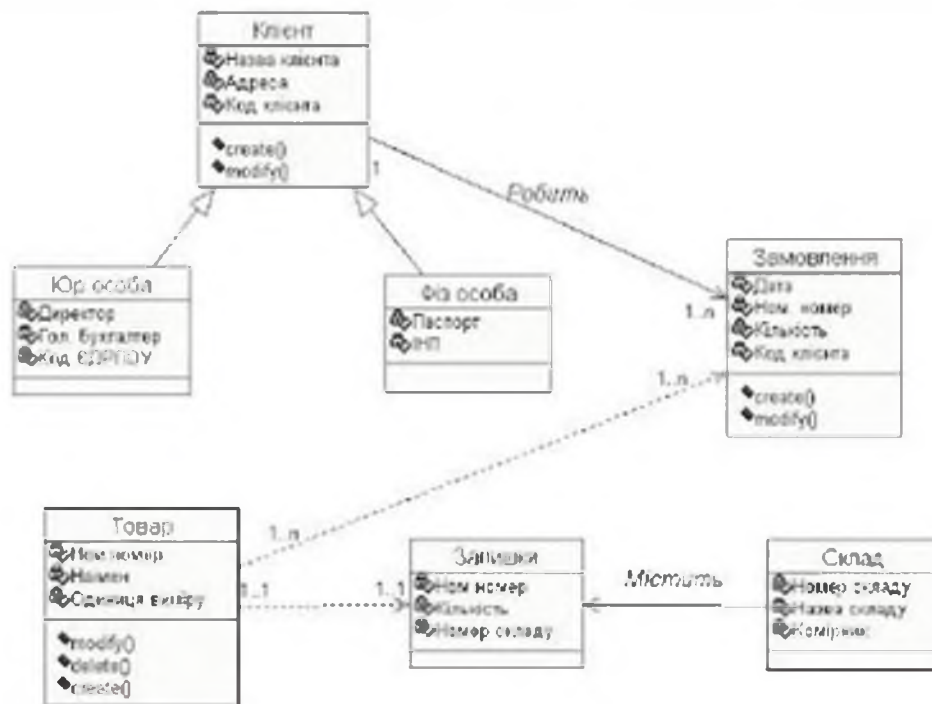
Інформаційна система аналізу стану особи.



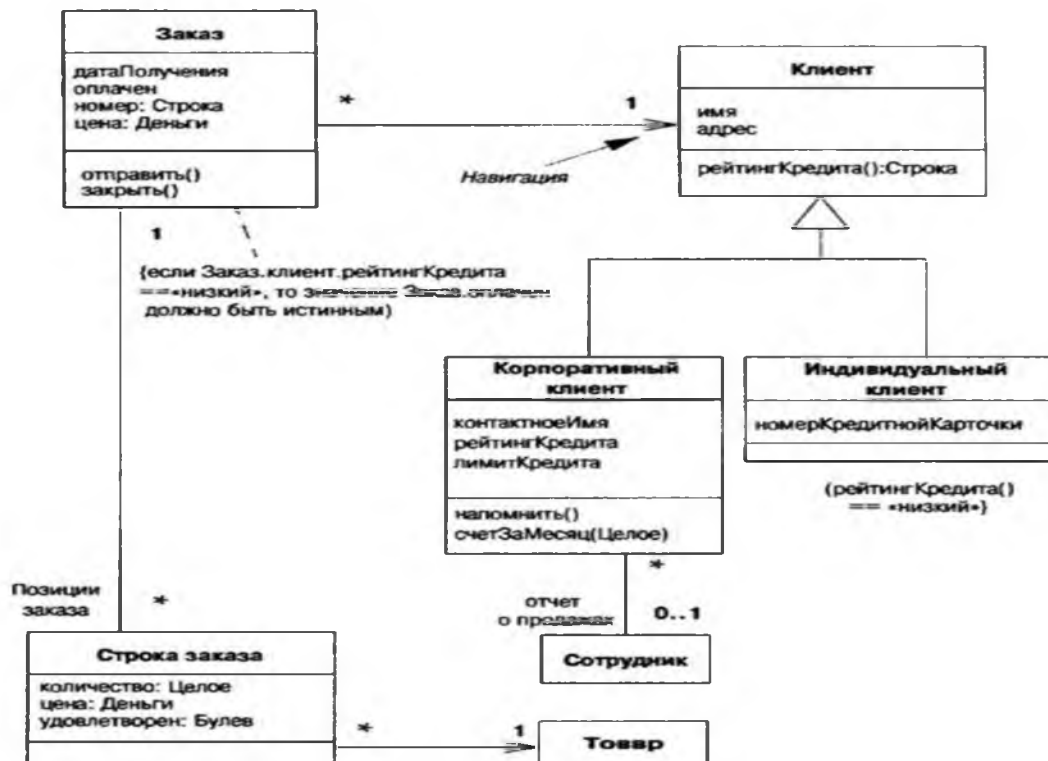
Приклади діаграм класів



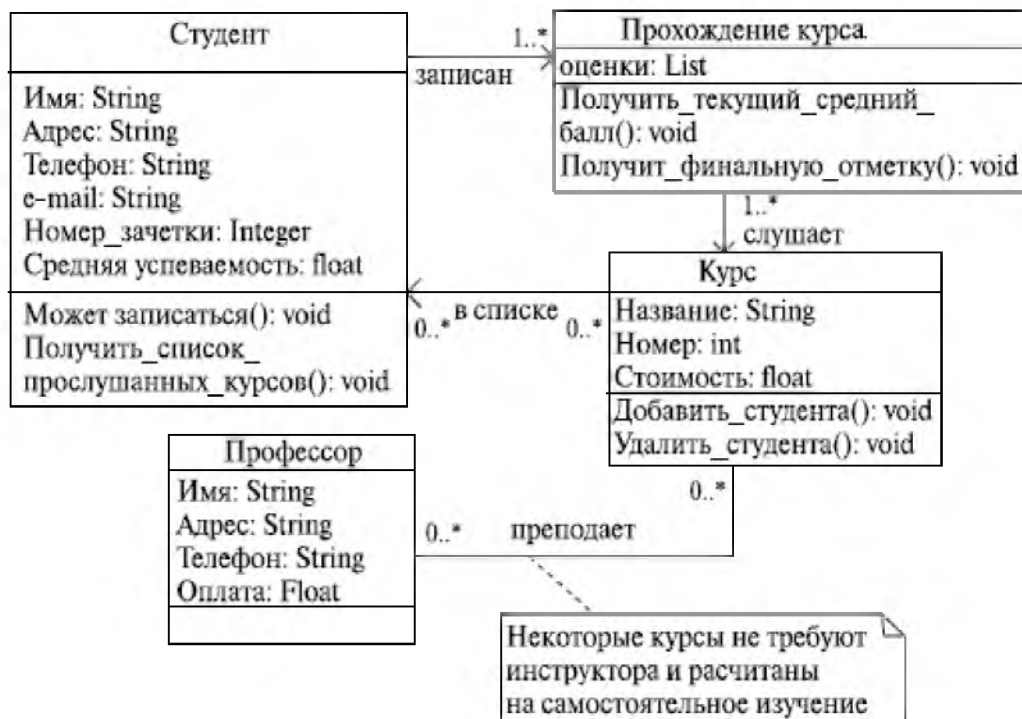
Замовлення товарів



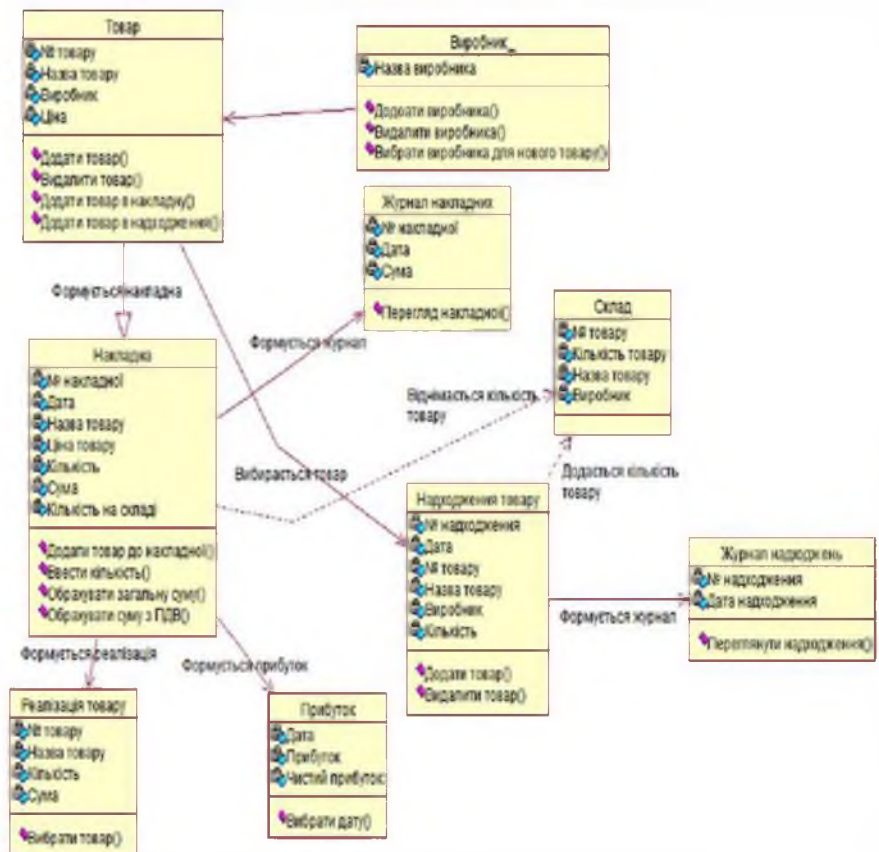
Заказ товаров



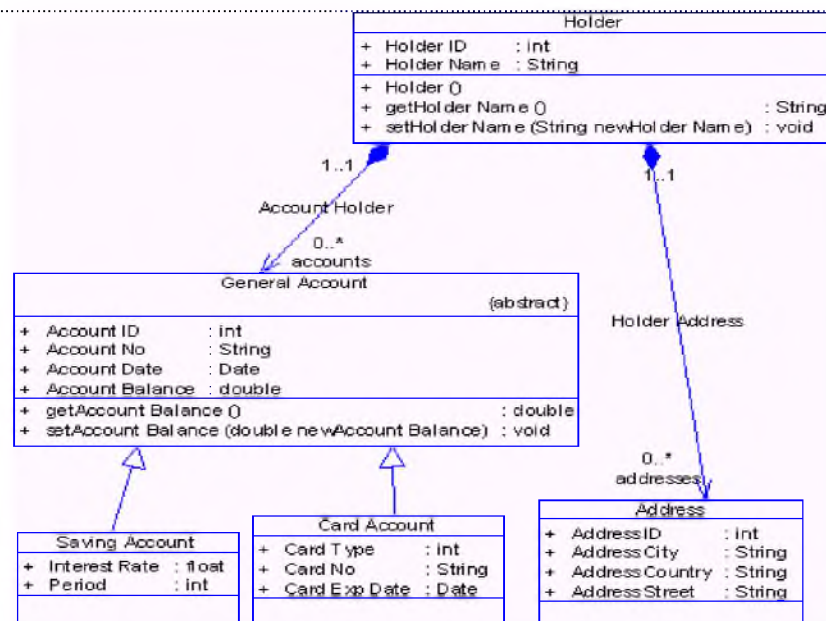
Облік студентів навчання студентів



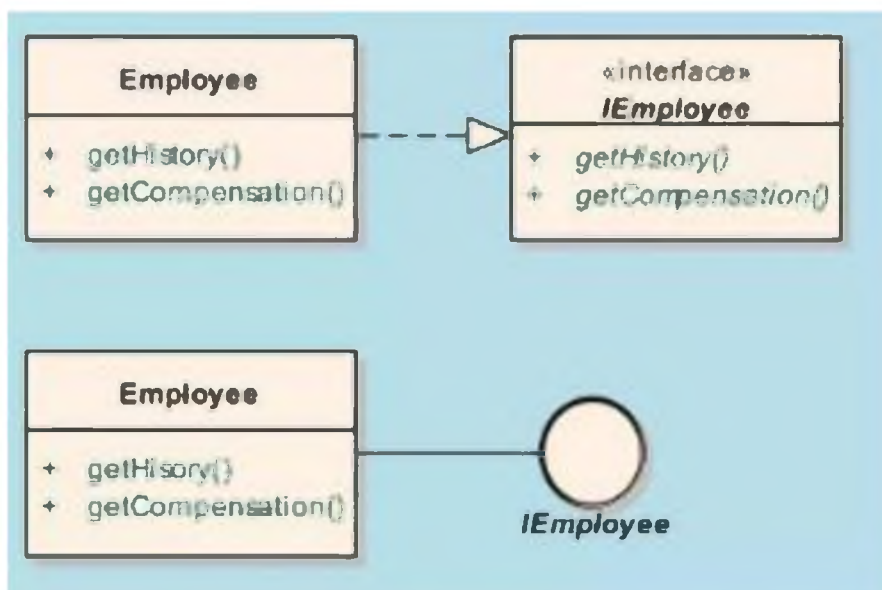
Визначення внутрішнього стану системи показується в моделі класів (class model). Клас (class) - це опис групи об'єктів із загальними властивостями (атрибутами), поведінкою (операціями), відносинами з іншими об'єктами і семантикою. Таким чином, клас є шаблоном для створення об'єкту. Зображена діаграма класів для системи «Магазин».



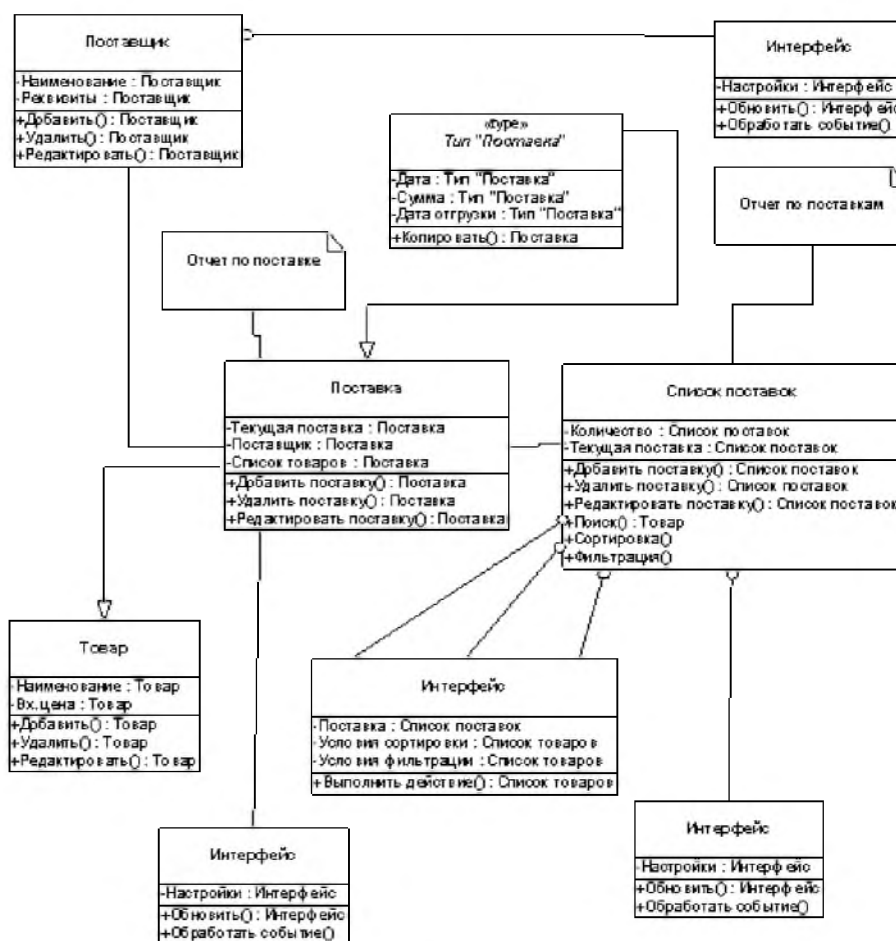
Оренда приміщень



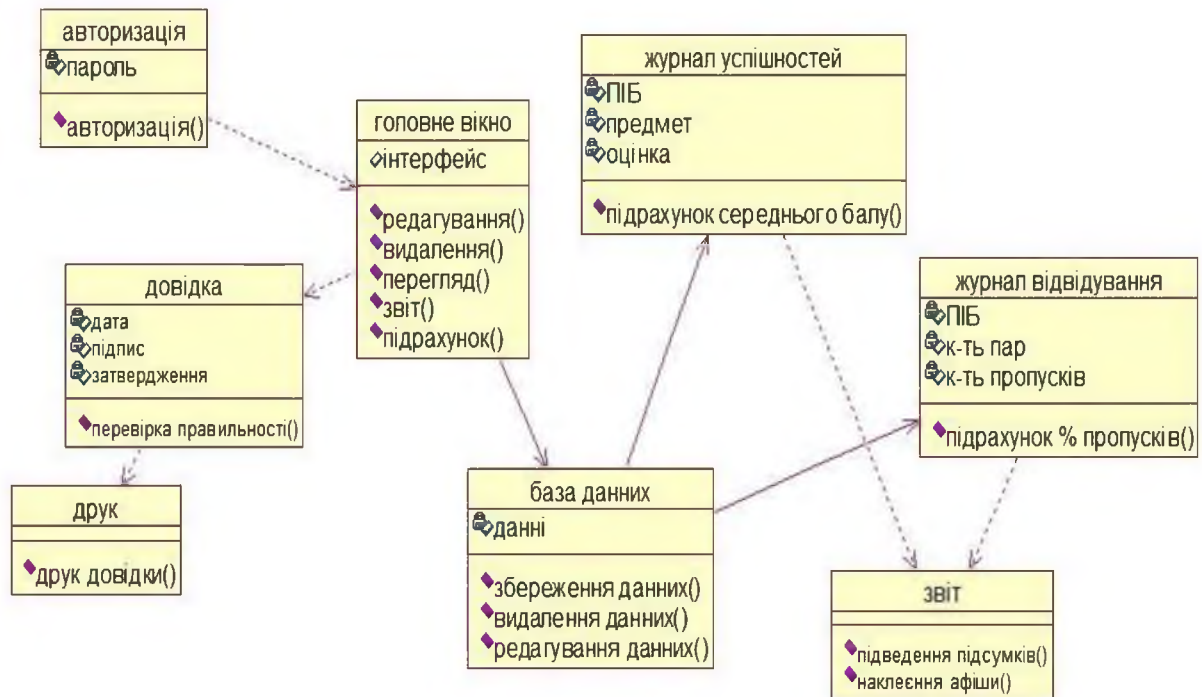
Виконавець



Поставка товару



Контроль успішності



Кадрова система

