**Практична робота №2 Заняття 1 Ознайомлення із засобами моделюванням програмного забезпечення за допомогою UML-діаграм.**

**Мета:** Ознайомитися з базовими поняттями CASE – засобів та мови UML.

**Завдання**:

**1)** Для виконання в подальшому лабораторних та практичних робіт визначитесь з вашими технічними можливостями.

Для побудови UML-діаграм пропонується безкоштовний інструмент Umbrello - UML Modeller, який можна завантажити за посиланням <https://umbrello.kde.org/>

Цей інструмент встановлюється на ОС Windows, починаючи з Windows 2007. До цього інструменту на сайті компанії надається україномовна документація, яку можна взяти на git-hub за посиланням <https://github.com/omc-college/ipz4-Software-Engineering> - файл Modelyuvannya-ta-analz-PZ-Pdruchnik-UML-Modeller.pdf. Більшість ілюстрацій до лекцій, практичних та лабораторних робіт з UML будуть надаватися за допомогою цього інструменту.

Для побудови UML-діаграм пропонується ряд безкоштовних інструментів за наданим нижче посиланням:

<https://www.quality-assurance-group.com/top-10-bezkoshtovnyh-onlajn-redaktoriv-dlya-stvorennya-uml-diagram-na-probu/> - перелік безкоштовних on-line редакторів .

Вам потрібно попередньо переглянути, щоб визначитись, який з них для вас зручніший.

Окрім того, UML-діаграми можна також будувати за допомогою Microsoft Office Visio 2010 і пізнішими версіями.

2) Ознайомитися з теоретичними відомостями.

1. Оформите звіт, в якому надайте перелік розглянутих вами інструментів побудови UML-діаграм з вашою характеристикою щодо можливості використання для вашого проекту.
2. Робота повинна бути виконана згідно критеріїв оформлення документації на аркушах формату А4.
3. Якщо потрібні пояснення – можна звернутись до викладача. Викладач сам вирішує відповідати на питання чи надати підказки, де можна знайти відповідь.
4. Протягом виконання лабораторної роботи студенти можуть використовувати конспекти або інші джерела інформації.
5. Роботу потрібно набирати на комп’ютері, що розташований в аудиторії або на власному ноутбукові, назвавши документ «Заняття з виконання практичної роботи№ ».
6. По закінченню практичну роботу потрібно здати на перевірку викладачеві, роздрукувавши її або надіславши електронною поштою. Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Результати надсилати на електронну адресу викладача [t.i.lumpova@gmail.com](mailto:t.i.lumpova@gmail.com)

Файл повинен мати назву в такому форматі:

**ОРІ<Номер групи><Номер лекції / практичної / лабораторної [літера позначення типу роботи L – лекція, P – практична, R – лабораторна]<Прізвище англійською>**. Наприклад, **ОРІ3101R**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт -"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи ІПЗ-31 - 26.02.2023**

**Розбір проблемних питань практичної роботи №1-4.**

1. Назвіть категорії фахівців, зайнятих розробкою ПЗ, на кожній фазі життєвого циклу програмного забезпечення (ЖЦ ПЗ). Надайте у вигляді:

назва *фази* ЖЦ ПЗ – перелік категорій.

В загальному випадку ми можемо визначити такі етапи, притаманні всім моделям:

**Специфікація** включає визначення вимог та їх аналіз;

**Розроблення** включає створення ПЗ відповідно до специфікацій (у т.ч. включає проектування)

**Тестування** – перевірка ПЗ на відповідність вимогам клієнта.

**Супровід/Модернізація** – розвиток ПЗ відповідно до змін потреб замовника.

На етапі розроблення завжди задіяні програмісти. Визначте ще інші категорії фахівців. Продумайте цю відповідь для інших етапів/фаз.

1. Наведіть приклади інструментів програмної інженерії.

Інструменти програмної інженерії поділяються на категорії, наприклад, інструменти розробки, інструменти тестування, інструменти управління конфігурацією. Microsoft Visual Studio є прикладом інструмента розробки. До якої ще категорії можна віднести Microsoft Visual Studio з погляду на можливості що він надає?

11. Наведіть приклади функціональних та нефункціональних вимог.

* ***Функціональні вимоги*** *– це перелік функцій або сервісів, які повинна надавати* система, а також обмежень на дані і поводження системи при їхньому виконанні. Наприклад, введення даних (наведіть ще інші).
* ***Нефункціональні вимоги*** *визначають умови виконання функцій (напр.,* захист інформації у БД, аутентифікація доступу до ПС тощо) у середовищі, що безпосередньо не пов'язані з функціями, а відбивають потреби користувачів щодо їх виконання. Наведіть ще інші приклади

**Теоретичні відомості.**

**Архітектура CASE – засобів**

Зазвичай до CASE - засобам відносять будь-який програмний засіб, що автоматизує ту або іншу сукупність процесів життєвого циклу програмного забезпечення і включає :

1. Репозиторій, що є основою CASE - засоби. Він є спеціалізованою базою даних проекту, призначеною для відображення стану проектованої інформаційної системи в кожен момент часу. Об'єкти усіх діаграм синхронізовані на основі загальної інформації словника даних. У репозиторії зберігаються описи наступних об'єктів : проектувальників і їх прав доступу до різних компонент системи; організаційних структур; діаграм; компонентів діаграм; зв'язків між діаграмами; структур даних; програмних модулів; процедур.

2. Графічний редактор, що забезпечує створення і редагування в заданій нотації і в інтерактивному (діалоговому) режимі елементів діаграм, зв'язків між ними, діаграм. У будь-який момент часу вони можуть бути роздруковані і включені в технічну документацію проекту.

3. Верифікатор діаграм (засіб тестування), що служить для контролю правильності побудови діаграм в заданій методології проектування. Його функції:

а) діагностика,

b) видача повідомлень про помилки,

с) виділення на діаграмі помилкових елементів.

4. Документатор проекту, що дозволяє отримувати інформацію про проекти у вигляді звітів. Звіти можуть будуватися:

а) за часом,

b) авторові,

с) елементам діаграм,

d) діаграмі,

е) проекту в цілому.

5. Адміністратор проекту, що виконує наступні функції :

а) ініціалізація (запуск),

b) завдання початкових параметрів проекту,

с) призначення і зміна прав доступу до елементів проекту.

6. Сервіс, що є набором системних утиліт по обслуговуванню репозиторія (архівація і відновлення даних, створення нового репозиторія).

На рис.1 показана архітектура CASE-засоби в загальному вигляді:

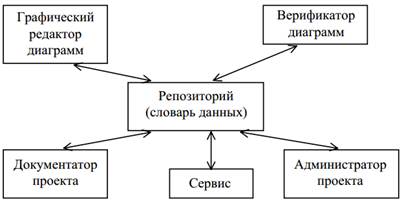


Рис.1. Архітектура CASE –засобів

CASE-технологія являє собою методологію проектування ІС, а також набір інструментальних засобів, що дозволяють у наочній формі моделювати предметну область, аналізувати цю модель на всіх етапах розробки і супроводу ІС і розробляти додатки відповідно до інформаційних потреб користувачів. Більшість існуючих CASE-засобів засновано на методологіях структурного (в-основному) або об'єктно-орієнтованого аналізу і проектування, що використовують специфікації у вигляді діаграм або текстів для опису зовнішніх вимог, зв'язків між моделями системи, динаміки поводження системи та архітектури програмних засобів.

**Загальні відомості про мову UML**

Мова UML є мовою візуального моделювання, яка розроблена для специфікації, візуалізації, проектування і документування компонентів програмного забезпечення, бізнес-процесів і інших різних систем. Вона є простим і потужним засобом моделювання, який може бути ефективно використаний для побудови концептуальних, логічних і графічних моделей систем самого різного цільового призначення і різного ступеню складності. Мова UML заснована на деякому числі базових понять, які можуть бути вивчені і застосовані більшістю програмістів і розробників, знайомих з методами об'єктно-орієнтованого аналізу і проектування (ООАП).

Конструктивне використання мови UML базується на розумінні загальних принципів моделювання систем і особливостей процесу об'єктно-орієнтованого аналізу і проектування. Одним з основних принципів побудови моделей систем є принцип абстрагування, який вимагає включати в модель тільки ті аспекти проектованої системи, які мають безпосереднє відношення до виконуваних системою функцій, а усі другорядні деталі опускаються, щоб не ускладнювати процес аналізу і дослідження отриманої моделі.

Іншим принципом побудови моделей є принцип багатомодельності, який стверджує, що жодна єдина модель не може з достатньою мірою адекватності описувати різні аспекти достатньо складної системи. За методологією ООАП це означає, що достатньо повна модель системи допускає деяке число взаємозв'язаних представлень (views), кожне з яких адекватно відбиває деякий аспект поведінки або структури системи. При цьому найбільш загальними представленням системи прийнято вважати статичне і динамічне представлення, які, у свою чергу, можуть підлягати декомпозиції.

Ще одним принципом прикладного системного аналізу є принцип ієрархічної побудови моделей систем. Цей принцип розглядає процес побудови моделі на різних рівнях абстрагування або деталізації у рамках фіксованих представлень. При цьому початкова, або первинна, модель системи має найбільш загальне представлення (метапредставлення). Вона модель будується на початковому етапі проектування і може не містити багатьох деталей і аспектів модельованої системи.

Таким чином, процес ООАП можна представити як поетапний спуск від найбільш загальних моделей і представлень концептуального рівня до більш детальних представлень логічного і фізичного рівнів. При цьому на кожному з етапів ООАП ці моделі послідовно доповнюються все більшим числом деталей, що дозволяє їм адекватніше відбивати різні аспекти конкретної реалізації складної системи. Загальна схема взаємозв'язків моделей ООАП представлена на рис.2.

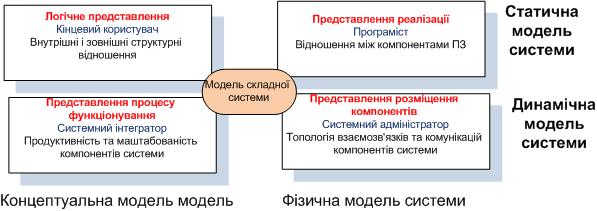


Рис. 2. Загальна схема взаємозв'язків моделей і представлень системи в процесі ООАП

Термін "фізична модель" в ООАП і мові UML має трактовку, що відрізняється від загальноприйнятої в загальній класифікації моделей систем.

У останньому випадку під фізичною моделлю системи розуміють деяку матеріальну конструкцію, що володіє властивостями подібності з формою оригіналу. Прикладами таких моделей можуть служити моделі технічних систем (літаків, кораблів), архітектурних споруд (будівель, мікрорайонів). В ООАП і мові UML фізична модель відбиває компонентний склад проектуємої системи з точки зору її реалізації на деякій технічній базі і обчислювальних платформах конкретних виробників.

Мова UML включає не лише абстрактні конструкції для представлення метамоделей систем, але і цілий ряд конкретних понять, що мають цілком визначену семантику. Це дозволяє мові UML одночасно досягти не лише універсальності представлення моделей для самих різних застосувань, але і можливості опису досить тонких де талів реалізації цих моделей стосовно конкретних систем.

**Загальна структура мови UML**

Універсальна мова моделювання (Unified Modelling Language або UML) — це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення. UML не є методом розробки, іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає вам наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками.

Розробкою UML керує Object Management Group (OMG). Ця мова є загальноприйнятим стандартом графічного опису програмного забезпечення.

UML розроблено для розробки структури зорієнтованого на об’єкти програмного забезпечення, ця мова має дуже обмежену користь для програмування на основі інших парадигм.

З найзагальнішої точки зору опис мови UML складається з двох взаємодіючих частин: семантики і нотації.

Семантикою мови UML є деяка метамодель, яка визначає абстрактний синтаксис і семантику понять об'єктного моделювання на мові UML.

Нотація мови UML є графічною нотацією і для візуального представлення семантики мови UML.

Семантика визначається для двох видів об'єктних моделей: структурних моделей і моделей поведінки. Структурні моделі, звані також статичними, описують структуру сутностей або компонентів деякої системи, включаючи їх класи, інтерфейси, атрибути і стосунки. Моделі поведінки, звані іноді динамічними, описують поведінку функціонування об'єктів системи, включаючи їх методи, взаємодію і співпраця між ними, а також процес зміни станів окремих компонентів і системи в цілому.

Формальний опис самої мови UML базується на деякій загальній ієрархічній структурі модельних уявлень, що має чотири рівні: метаметамодель; метамодель; модель; об'єкти користувача.

Рівень метаметамоделі утворює початкову основу для усіх метамодельних представлень. Головне призначення цього рівня полягає в тому, щоб визначити мову для специфікації метамоделі.

Метамодель є екземпляром або конкретизацією метаметамодели. Головне завдання цього рівня - визначити мову для специфікації моделей. Усі основні поняття мови UML - це поняття рівня метамоделі. Приклади таких понять: клас, атрибут, операція, компонент, асоціація і багато інших.

Представлення інформаційної системи на мові UML :

1. Представлення використання - основна частина моделі опису системи.

2. Логічне представлення - опис функціональних можливостей системи.

3. Компонентне представлення - опис структури і взаємозв'язків модулів системи.

4. Представлення взаємодії процесів - опис погоджених дій модулів системи.

5. Представлення розподілу - опис фізичної архітектури системи.

Ці представлення надаються через такий базовий набір моделей:

*Варіантів використання (use case diagram);*

*Класів (class diagram);*

*Кооперації (collaboration diagram);*

*Послідовності (sequence diagram);*

*Станів (statechart diagram);*

*Діяльності (activity diagram);*

*Компонентів (component diagram);*

*Розгортання (deployment diagram).*

**Принципи моделювання в UML**

Використання мови UML базується на таких загальних принципах моделювання:

* ***абстрагування*** – у модель потрібно включати тільки ті елементи проектованої системи, які мають безпосереднє відношення до виконання їй своїх функцій або свого цільового призначення. Інші елементи опускаються, щоб не ускладнювати процес аналізу та дослідження моделі;
* ***многомодельность*** – ніяка єдина модель не може з достатнім ступенем точності описати різні аспекти системи. Допускається описувати систему деяким числом взаємозалежних уявлень, кожне з яких відображає певний аспект її поведінки або структури;
* ***ієрархічна побудова*** – при описі системи використовуються різні рівні абстрагування і деталізації в рамках фіксованих уявлень. При цьому перше представлення системи описує її в найбільш загальних рисах і є поданням концептуального рівня, а наступні рівні розкривають різні аспекти системи з зростаючої ступенем деталізації аж до фізичного рівня. Модель фізичного рівня в UML відображає компонентний склад проектованої системи з точки зору її реалізації на апаратурної і програмної платформах конкретних виробників.

Спільно ці три механізми розширення мови дозволяють модифікувати його відповідно до потреб проекту або особливостями технології розробки.

**Сутності в UML**

У UML визначено чотири типи сутностей: структурні, поведінкові, групуючи і анотаційні. Сутності є основними об'єктно-орієнтованими елементами мови, за допомогою яких створюються моделі.

**Структурні сутності** – це імена іменники в моделях на мові UML. Як правило, вони представляють статичні частини моделі, відповідні концептуальним або фізичним елементам системи. Прикладами структурних сутностей є «клас», «інтерфейс», «кооперація», «прецедент», «компонент», «вузол», «актор».

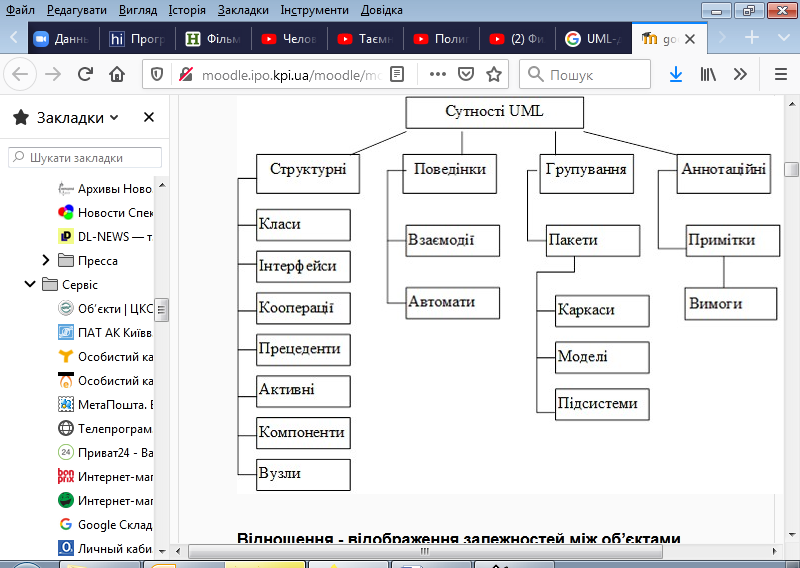
**Поведінкові сутності** є динамічними складовими моделі UML. Це дієслова, які описують поведінку моделі в часі і в просторі. Існує два основних типи поведінкових сутностей:

* ***взаємодія*** – це поведінка, суть якого полягає в обміні повідомленнями між об'єктами в рамках конкретного контексту для досягнення певної мети;
* ***автомат*** – алгоритм поведінки, що визначає послідовність станів, через які об'єкт або взаємодія проходять у відповідь на різні події.

**Групуючи сутності** є організуючими частинами моделі UML. Це блоки, на які можна розкласти модель. Така первинна сутність мається на єдиному екземплярі – це пакет.

***Пакети*** становлять собою універсальний механізм організації елементів у групи. У пакет можна помістити структурні, поведінкові й інші групуючи сутності. На відміну від компонентів, які реально існують під час роботи програми, пакети носять чисто концептуальний характер, тобто існують тільки в процесі розробки.

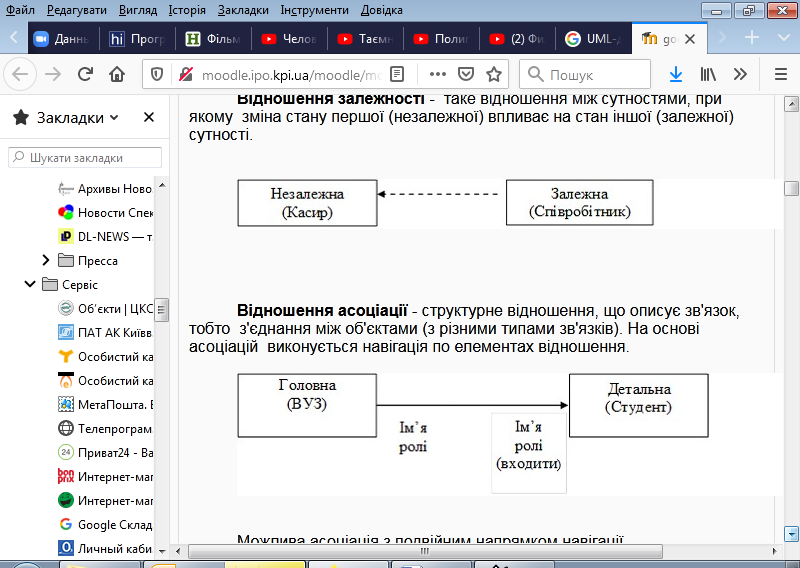
**Анотаційні сутності** – Це пояснювальні частини моделі UML: коментарі для додаткового опису, роз'яснення чи зауваження до будь-якого елементу моделі. Є тільки один базовий тип анотаційних елементів – ***примітка***. Примітку використовують, щоб забезпечити діаграми коментарями чи обмеженнями, вираженими у вигляді неформального або формального тексту.

****

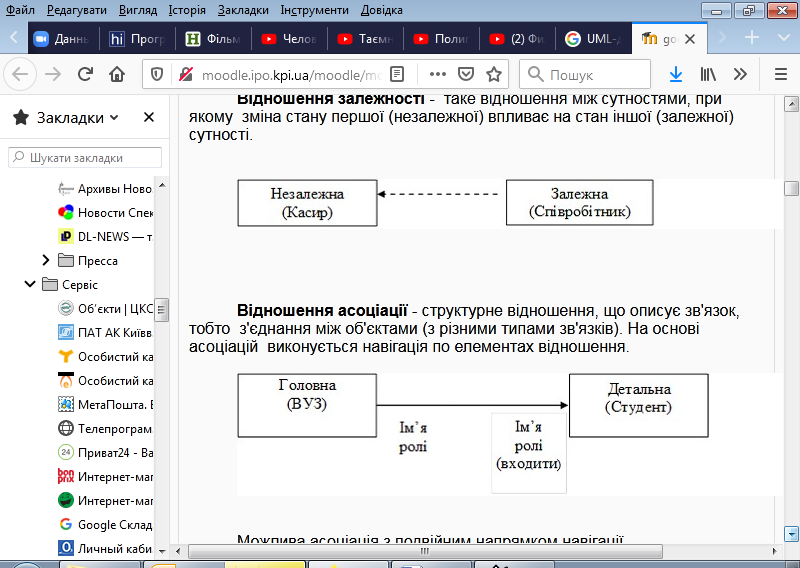
**Відносини в UML**

У мові UML визначено такі типи відносин: залежність, асоціація, узагальнення і реалізація. Ці відносини є основними єднальними конструкціями UML і також як сутності застосовуються для побудови моделей.

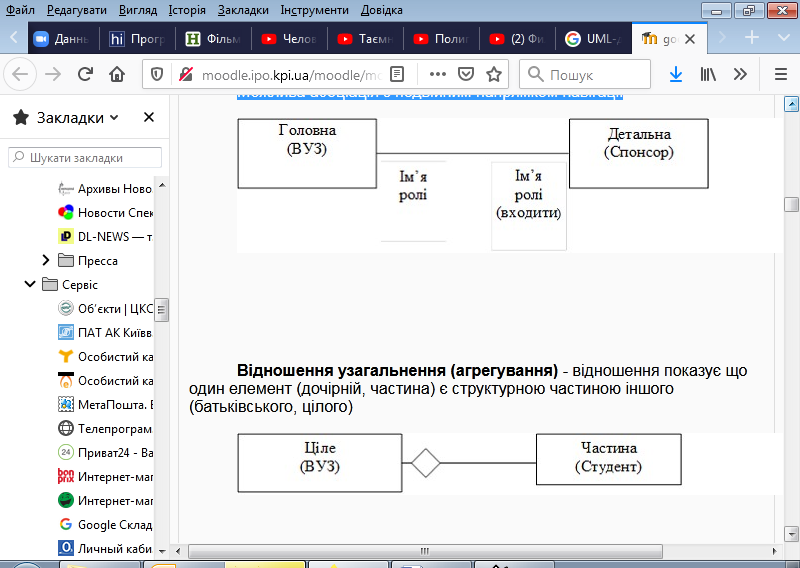
***Залежність*** (Dependency) – це семантичне відношення між двома сутностями, при якому зміна однієї з них, незалежної, може вплинути на семантику іншого, залежною.



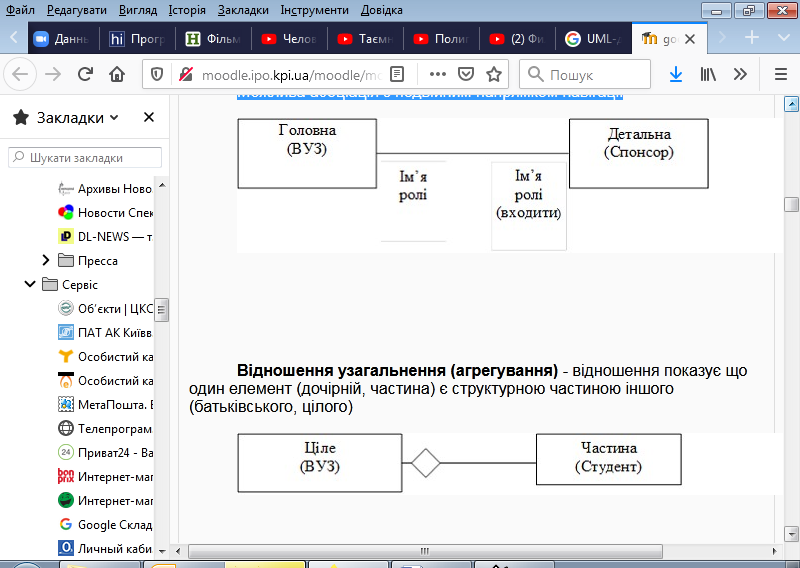
***Асоціація*** (Association) – структурне сполучення, що описує сукупність смислових або логічних зв'язків між об'єктами.

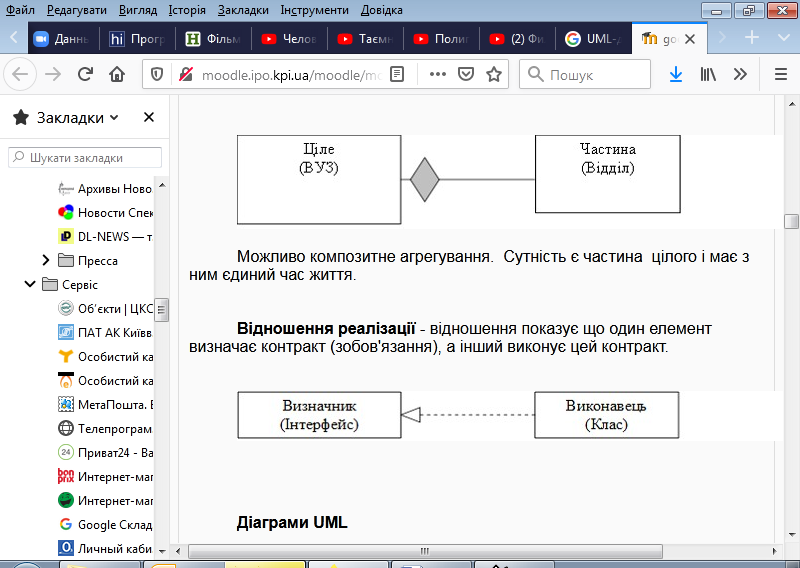


Можлива асоціація з подвійним напрямком навігації



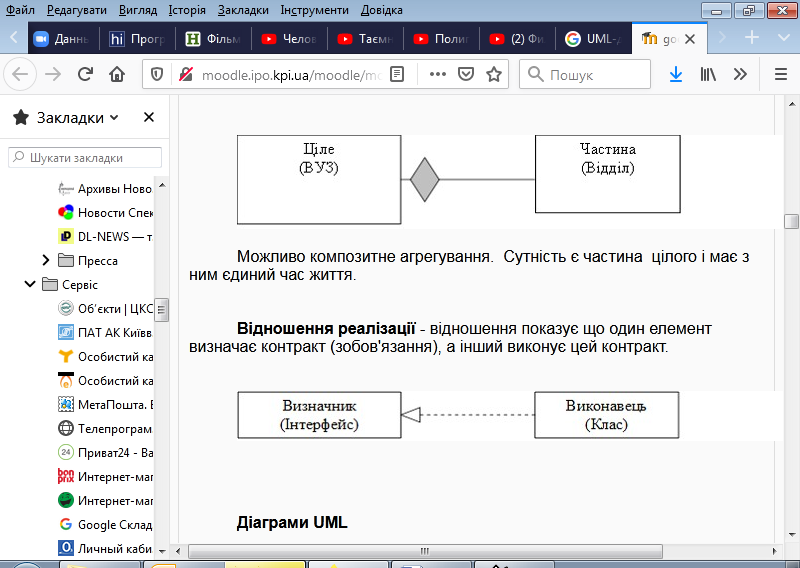
***Узагальнення*** (Generalization) або агрегування – це відношення, при якому об'єкт спеціалізованого елемента (нащадок) може бути підставлений замість об'єкта узагальненого елемента (предка). При цьому, відповідно до принципів об'єктно-орієнтованого програмування, нащадок (child) успадковує структуру і поведінку свого предка (parent).





***Реалізація*** (Realization) є семантичним відношенням між класифікаторами, при якому один класифікатор визначає зобов'язання, а інший гарантує його виконання, тобто відношення реалізації показує, що один елемент визначає контракт (зобов'язання), а інший виконує цей контракт.. Ставлення реалізації зустрічаються у двох випадках:

* між інтерфейсами і реалізують їх класами чи компонентами;
* між прецедентами і реалізують їх кооперації.



**Загальні механізми UML**

Для точного опису системи в UML використовуються, так звані, загальні механізми :

* специфікації (specifications);
* доповнення (adornments);
* ділення (common divisions);
* розширення (extensibility mechanisms).

UML є не тільки графічною мовою. За кожним графічним елементом його нотації стоїть специфікація, що містить текстове представлення відповідної конструкції мови. Наприклад, піктограмі класу відповідає специфікація, яка описує його атрибути, операції та поведінку, хоч візуально, на діаграмі, піктограма часто відображає тільки малу частину цієї інформації. Більш того, в моделі може бути присутнім інше представлення цього класу, що відбиває зовсім інші його аспекти, але, тим не менш, відповідне специфікації. Таким чином, графічна нотація UML використовуються для візуалізації системи, а з допомогою специфікацій описують її деталі.

Практично кожен елемент UML має унікальне графічне зображення, яке дає візуальне уявлення найважливіших його характеристик. Нотація сутності «клас» містить його ім'я, атрибути та операції. Специфікація класу може містити й інші деталі, наприклад, видимість атрибутів і операцій, коментарі або вказівка на те, що клас є абстрактним. Багато хто з цих деталей можна візуалізувати у вигляді графічних або текстових доповнень до стандартного прямокутника, який зображує клас.

При моделюванні об'єктно-орієнтованих систем існує певний поділ експонованих сутностей.

По-перше, існує поділ на класи та об'єкти. Клас – це абстракція, а об'єкт – конкретне втілення цієї абстракції. У зв'язку з цим, практично всі конструкції мови характеризуються подвійністю «Клас / об'єкт». Так, є прецеденти та примірники / екземпляри прецедентів, компоненти і примірники компонентів, вузли та примірники вузлів. У графічному поданні для об'єкта прийнято використовувати той же символ, що і для класу, а назву підкреслювати.

По-друге, існує розподіл на інтерфейс і його реалізацію. Інтерфейс декларує зобов'язання, а реалізація представляє конкретне втілення цих зобов'язань та забезпечує точне проходження оголошеної семантики. У зв'язку з цим, майже всі конструкції UML характеризуються подвійністю «інтерфейс / реалізація». Наприклад, прецеденти реалізуються через кооперації, а операції – методами.

UML є відкритим мовою, тобто допускає контрольовані розширення , Щоб відобразити особливості моделей предметних областей. Механізми розширення UML включають:

* стереотипи (Stereotype) – розширює словник UML, дозволяючи на основі існуючих елементів мови створювати нові, орієнтовані на вирішення конкретної проблеми;
* помічені значення (Tagged value) – розширює властивості основних конструкцій UML, дозволяючи включати додаткову інформацію в специфікацію елемента;
* обмеження (Constraints) – розширює семантику конструкцій UML, дозволяючи створювати нові і скасовувати існуючі правила.

Спільно ці три механізми розширення мови дозволяють модифікувати його відповідно до потреб проекту або особливостями технології розробки.