Що таке об'єктно-орієнтоване програмування

Кафедра комп’ютерних наук. Ст. викл. Міловідов Ю.О.

Протягом всієї історії програмування ускладнення програм змушувало програмістів шукати шляхи, які б дозволили впоратися зі складністю.

Програмування для перших обчислювальних машин полягало в перемиканні перемикачів на їх передній панелі таким чином, щоб їх положення відповідало двійковим кодами машинних команд.

Оскільки програми продовжували рости в розмірах, бажання впоратися з більш високим рівнем складності викликало появу мов високого рівня, розробка яких дала програмістам більше інструментів.

Програмне забезпечення (ПЗ) як складна система

Головна риса промислового ПЗ – *рівень складності*: один розробник практично не в змозі охопити всі аспекти системи. Рівень складності промислового ПЗ перевищує можливості людського інтелекту.

Промислові програмні продукти характеризуються:

- Великою тривалістю життєвого циклу,
- Великою кількістю користувачів, які сильно залежать від нормального функціонування системи.

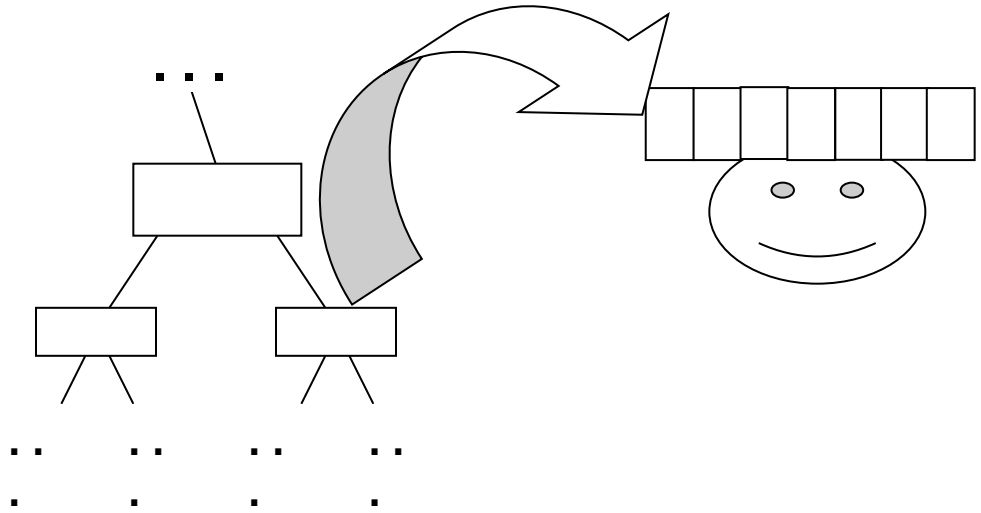
Приклади:

- Системи керування складними технічними об’єктами (літак, корабель, електростанція).
- Системи за діяльністю людини (диспетчеризація повітряного та залізничного транспорту; система обліку та управління підприємством (класу ERP)).
- Система управління базою даних (сотні тисяч і мільйони записів, десятки одночасних оновлень).

Роль декомпозиції в боротьбі зі складністю

„Спосіб управління складними системами був відомий ще з давніх часів – *divide et imperia* (розділяй і володарюй).” – Дейкстра.

У процесі проектування складної програмної системи необхідно поділяти її на все менші та менші підсистеми, кожну з яких можна вдосконалювати незалежно.

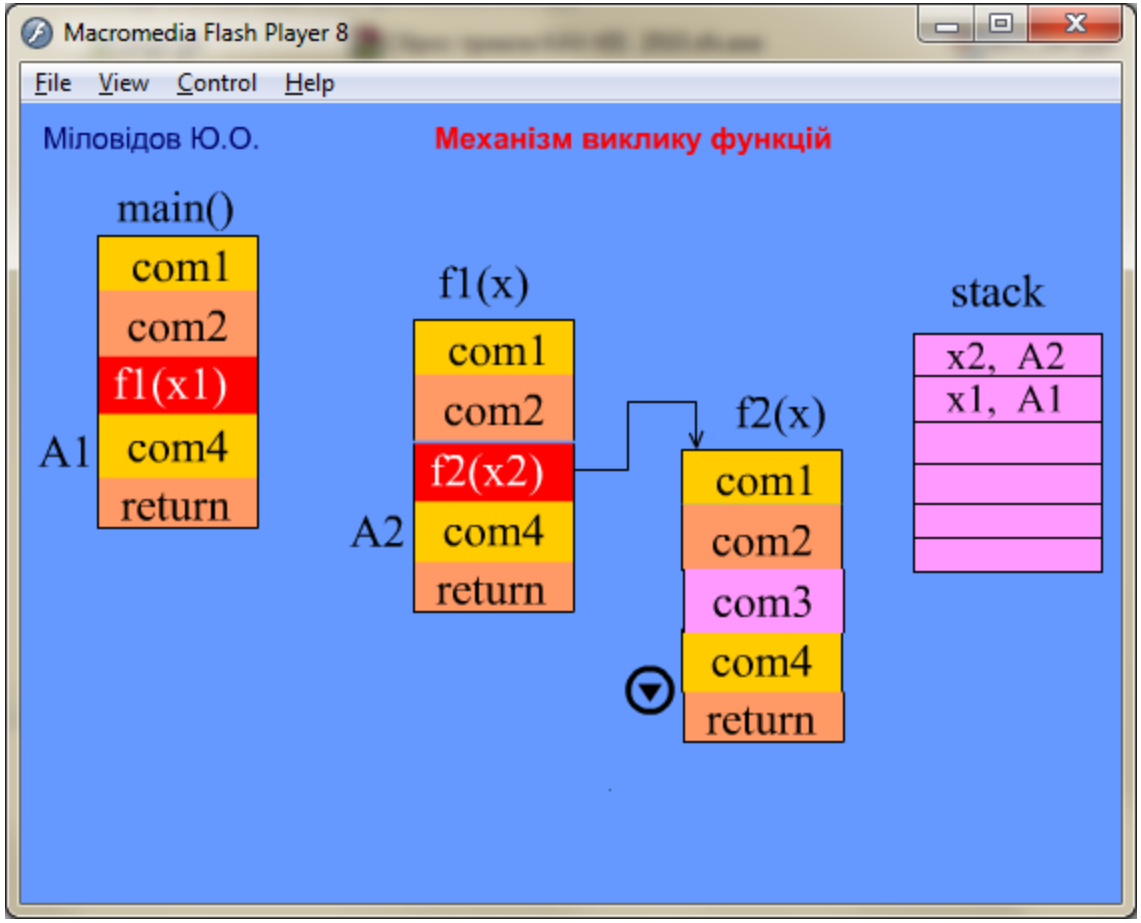


Для розуміння будь-якого рівня системи нам треба одночасно тримати в голові лише кілька її частин

Спосіб управління складними системами відомий вже дуже давно: "Розділяй і володарюй". Поділ програми на дрібні шматочки, кожен з яких може працювати незалежно, - ось перший крок в боротьбі зі складністю і називається декомпозицією.

У структурному програмуванні під декомпозицией розумілося розділення алгоритмів, де кожен алгоритм виконував один з етапів загального процесу. Основою цього принципу є проектування "зверху вниз".

Алгоритмічна декомпозиція



Об'єктно-орієнтоване програмування об'єднало кращі ідеї структурованого з рядом потужних концепцій, які сприяють більш ефективної організації програм. Об'єктно-орієнтований підхід до програмування дозволяє розкласти завдання на складові частини таким чином, що кожна складова частина буде являти собою самостійний об'єкт, який містить власні інструкції і дані. При такому підході істотно знижується загальний рівень складності програм, що дозволяє програмісту справлятися з більш складними програмами, ніж раніше (тобто написаними при використанні структурованого програмування).

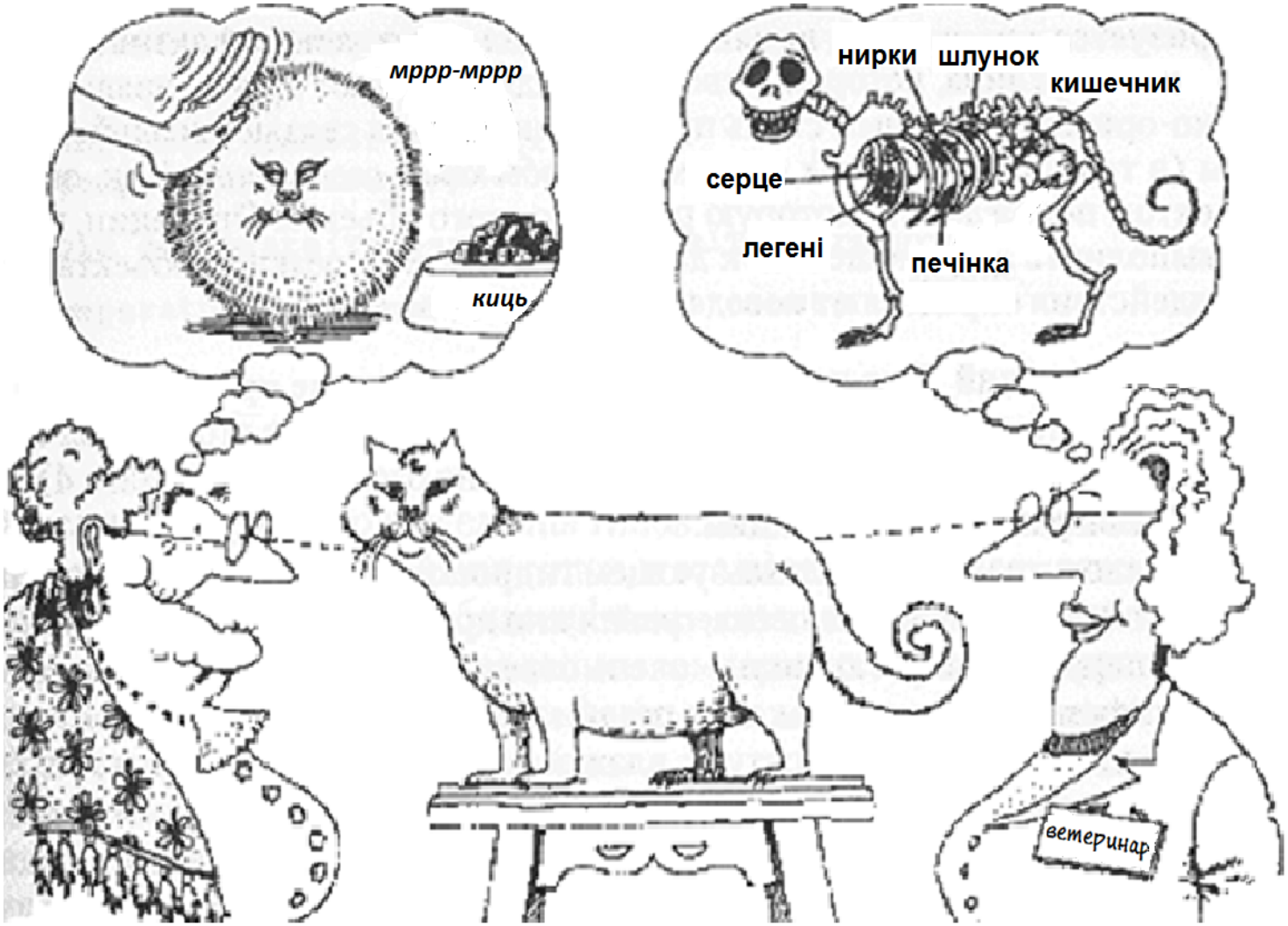
Об’єктно-орієнтована декомпозиція

Nature -> Animal -> Cat

- кожний об’єкт має власну поведінку,
- та кожний об’єкт моделює певний об’єкт реального світу.

Абстракція

У людини є засіб, що дозволяє боротися зі складністю і він з’явився задовго до появи перших комп'ютерів. Люди абстрагувалися від складності. Не маючи можливості обійняти неосяжне, вони ігнорували дрібні і не важливі деталі, зосередивши свою увагу на основних, ключових моментах. Ми намагаємося створити якусь узагальнену модель дійсності. Об'єкти, отримані в результаті декомпозиції, являють собою абстракції реального світу. Абстракція фокусує увагу на суттєві характеристики об'єкта



Абстракція фокусує увагу на суттєві характеристики об'єкта з точки зору спостерігача

Ієрархія

Багато об'єктів мають загальну поведінку або схожу на поведінку інших об'єктів. Розділивши об'єкти на групи, виділивши найбільш загальну модель поведінки і залишаючи спеціалізовану поведінку на деякі виділені об'єкти, ми сформуємо ієрархію об'єктів, що значно спрощує розуміння складної системи.

Об'єктно-орієнтований підхід

Об'єктно-орієнтована технологія грунтується на так званій об'єктній моделі. Основними властивостями цієї моделі є: абстрагування, інкапсуляція, модульність, ієрархічність, типізація, паралелізм і збереженість. Це не нові принципи, але в сукупності вони зустрічаються тільки в об'єктному моделюванні.

Об'єктна модель є основою трьох китів розробки: об'єктно-орієнтованого аналізу, об'єктно-орієнтованого проектування та об'єктно-орієнтованого програмування.

Наш курс присвячений третьому (останньому) киту – об'єктно-орієнтованому програмуванню.

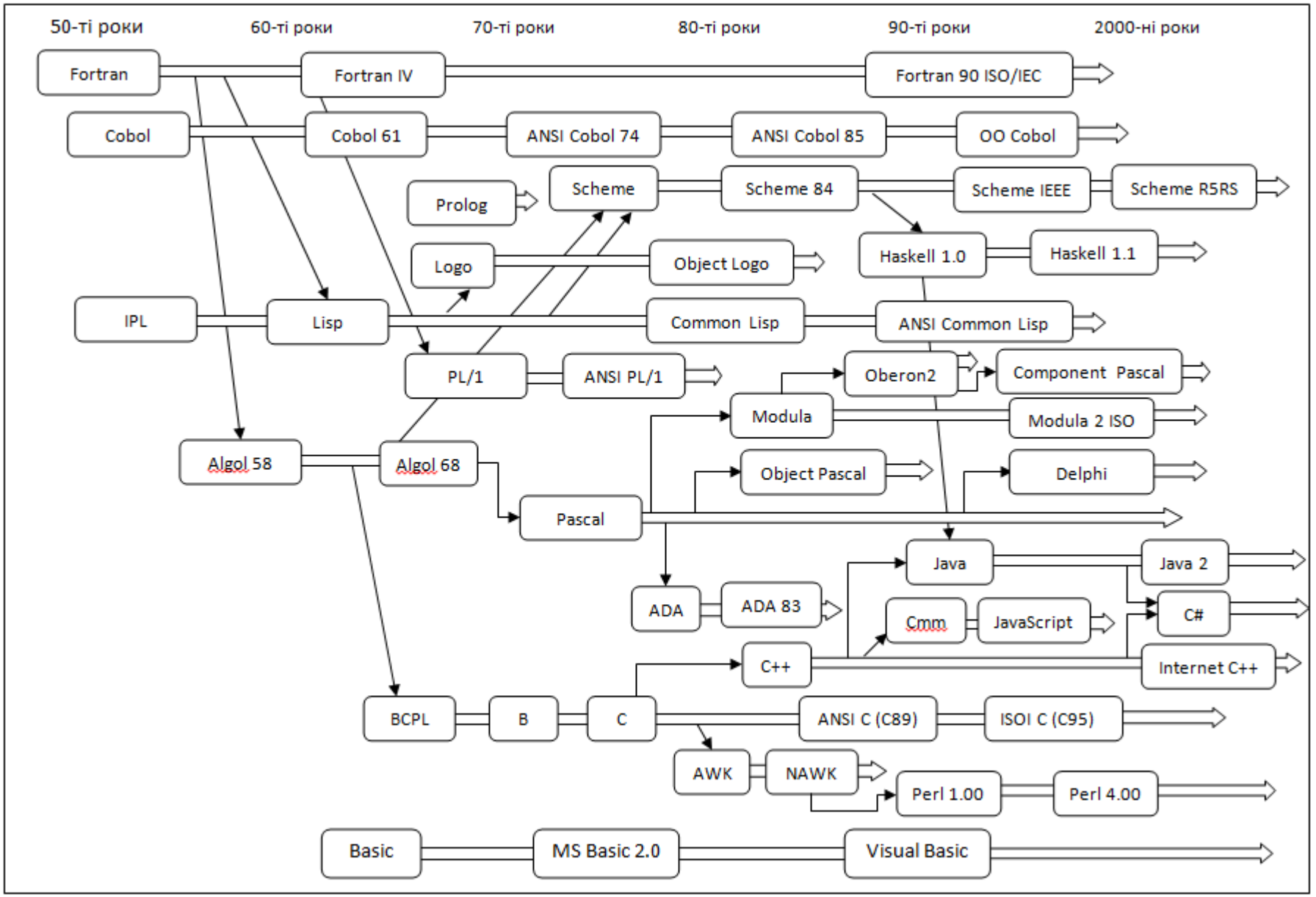
Нова парадигма

Об'єктно-орієнтоване програмування (ООП) стало основною методологією програмування 90-х років. ООП є продуктом 25-річної практики і включає ряд мов: Lisp, Clu, Actor, Eiffel, Objective C, C++, Java, C#, Python та інші. Це техніка програмування, яка дозволяє фіксувати поведінку реального світу таким способом, при якому деталі реалізації приховані. ООП часто називають новою парадигмою програмування. Інші відомі парадигми програмування: директивна (структурне програмування - Pascal, C), логічна – Prolog, функціональна – Lisp, Effel. Парадигми в програмуванні визначають, як проводити обчислення, як повинна бути структурована і організована робота комп'ютера.

Мова програмування

Мова, якою ми розмовляємо, безпосередньо впливає на спосіб сприйняття світу. Це стосується не тільки природних мов, але і штучних мов – мов програмування. Щоб ефективно використовувати ООП, потрібно дивитися на світ іншим способом: не з точки зору структурних мов.

Мова програмування, яка претендує називатися об'єктно-орієнтованою, повинна надавати зручний механізм підтримки об'єктно-орієнтованого стилю програмування.



Генеалогічне дерево мов програмування

Не слід говорити про те, яка мова краща за іншу. Можна лише відзначити, чи є в мові засоби, достатні для підтримки стилю програмування. Ми розглянемо об'єктно-орієнтований стиль програмування на прикладі двох мов: C++ і C#.

Мова C++ – це ключ до сучасного об'єктно-орієнтованого програмування. Вона створена для розробки високопродуктивного програмного забезпечення і надзвичайно популярна серед програмістів. Сьогодні бути професійним програмістом високого класу означає бути компетентним в C++.

Ця мова не просто популярна. Вона забезпечує концептуальний фундамент, на який спираються інші мови програмування і багато сучасних засобів обробки даних. Адже не випадково нащадками C++ стали такі шановані мови, як Java і C#.

Основним поняттям C# є об'єктно-орієнтоване програмування. Методика ООП невіддільна від C#, і тому всі програми на C# є об'єктно-орієнтованими хоча б в найменшій мірі.

У зв'язку з цим дуже важливо і корисно засвоїти основні принципи ООП, перш ніж приступати до написання найпростішої програми на C#.

У першій частині курсу розглянути базові принципи ООП і об’єктно-орієнтована технологія програмування на мові C++, у другій частині – на мові C#.

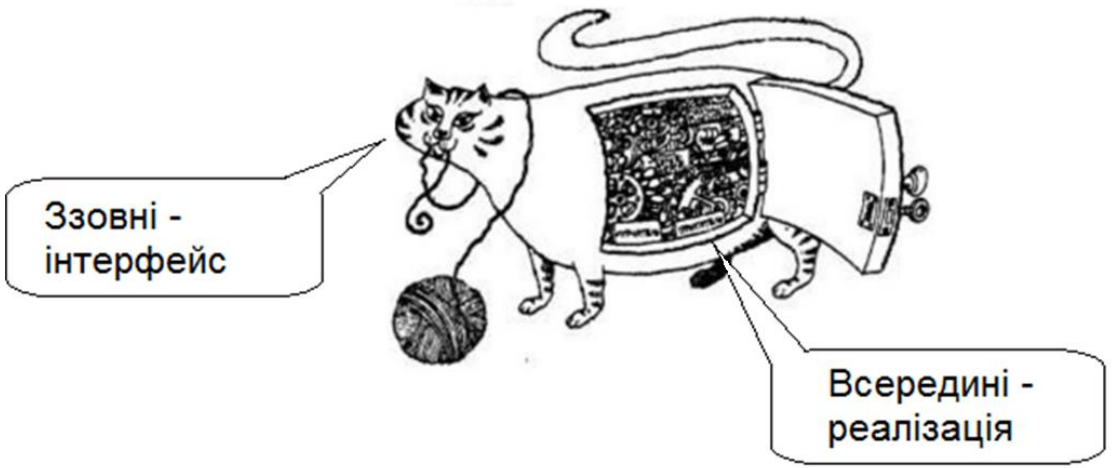
Принципи об'єктно-орієнтованого програмування

Всі мови об'єктно-орієнтованого програмування характеризуються трьома загальними ознаками: **інкапсуляцією, спадкуванням і поліморфізмом.**

Інкапсуляція

У об'єктно-орієнтованих системах дані комбінуються з конкретною поведінкою, тобто з діями, здійснюваними над ними. Все це об'єднується в клас.

При цьому приховуються деталі реалізації методів класу, залишаючи назовні лише взаємодію через інтерфейс.



Програми, як правило, складаються з двох основних елементів: інструкцій (коду) і даних. Код - це частина програми, яка виконує дії, а дані представляють собою інформацію, на яку спрямовані ці дії. Інкапсуляція - це такий механізм програмування, який пов'язує воєдино код і дані, які він обробляє, щоб убезпечити їх як від зовнішнього втручання, так і від неправильного використання.

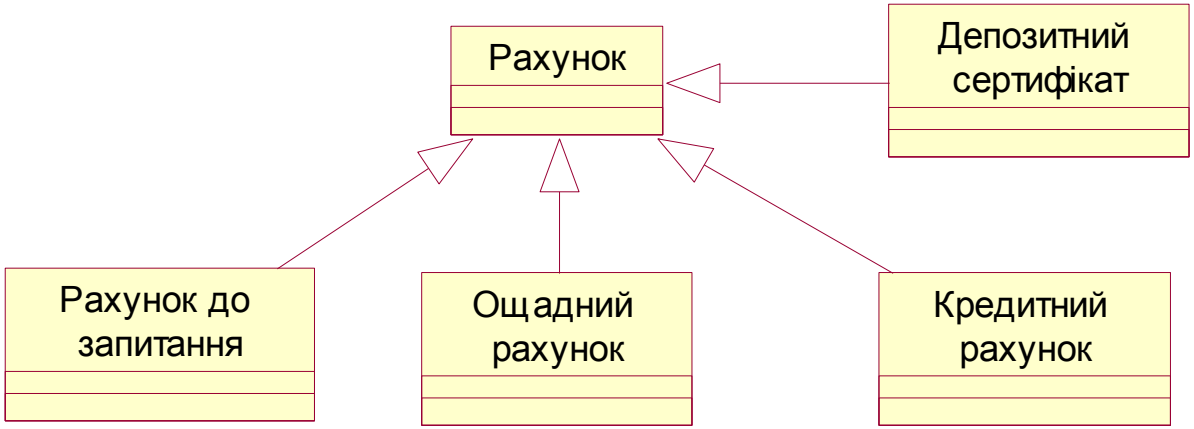
В об'єктно-орієнтованої мови код і дані можуть бути пов'язані способом, при якому створюється самостійний чорний ящик. У цьому "ящику" містяться всі необхідні (для забезпечення самостійності) дані і код. При такому зв'язуванні коду і даних створюється об'єкт, тобто об'єкт - це конструкція, яка підтримує інкапсуляцію.

Усередині об'єкту, код, дані або обидві ці складові можуть бути закритими в "рамках" цього об'єкта або відкритими. Закритий код (або дані) відомий і доступний тільки іншим частинам того ж об'єкта. Іншими словами, до закритого коду або даних не може отримати доступ та частина програми, яка існує поза цим об'єкта. Відкритий код (або дані) доступний будь-яким іншим частинам програми, навіть якщо вони визначені в інших об'єктах. Зазвичай відкриті частини об'єкта використовуються для надання керованого інтерфейсу з закритими елементами об'єкта.

Спадкування

Спадкування – це процес, завдяки якому один об'єкт може набувати властивості іншого. Наприклад, кіт Томас є частиною сімейства котячих, яке в свою чергу є частиною класу ссавці, а той – частиною ще більшого класу тварини.

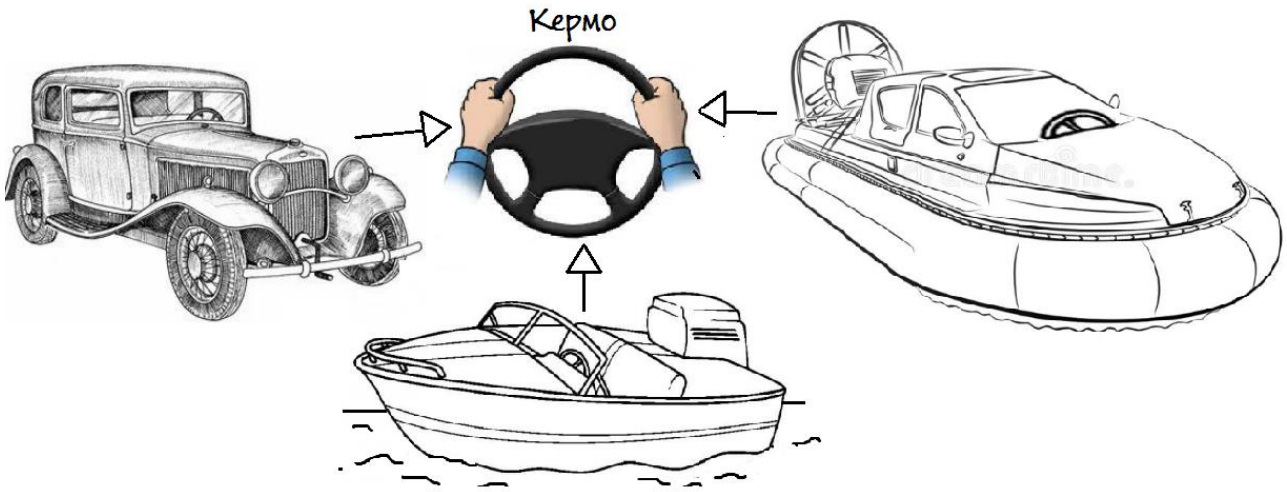
У об'єктно-орієнтованих системах успадкування є механізм, що дозволяє створювати нові об'єкти, грунтуючись на тих, що вже існують. Породжуваний (child) об'єкт-нащадок успадковує властивості батьківського (parent) об'єкту, що його породжує.



Завдяки наслідуванню об'єкту потрібно визначити тільки ті якості, які роблять його унікальним всередині його класу, оскільки він (об'єкт) успадковує загальні атрибути свого батька.

Поліморфізм

Поліморфізм (від грецького слова polymorphism, що означає "багато форм") - це властивість, що дозволяє використовувати один інтерфейс для цілого класу дій. Конкретна дія визначається характерними ознаками ситуації. Як простий приклад поліморфізму можна привести кермо автомобіля.



Для керма (тобто інтерфейсу) байдуже, який тип рульового механізму використовується в автомобілі. Якщо ви знаєте, як поводитися з кермом, ви зможете вести автомобіль будь-якого типу. Той же принцип можна застосувати до програмування.

У більш загальному вигляді концепція поліморфізму виражається фразою "один інтерфейс - багато методів". Це означає, що для групи пов'язаних дій можна використовувати один узагальнений інтерфейс. Поліморфізм дозволяє знизити рівень складності за рахунок можливості застосування одного і того ж інтерфейсу для завдання цілого класу дій.

Незважаючи на те, що ООП дійсно допомагає в проектуванні складних і великих програм, ООП не панацея. Щоб стати професіоналом в програмуванні, необхідний талант, здатність до творчості, інтелект, знання, логіка, уміння будувати і використовувати абстракції і досвід, навіть якщо використовуються кращі засоби розробки.

Дякую за увагу