Конспект лекції № 8-9-10

Тема № 5. КОНЦЕПЦІЯ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Міжпредметні зв'язки: Зв'язок із елементами знань і умінь таких навчальних дисциплін як „Алгоритмізація та програмування", „Інформатика", „Вступ до фаху", „Технології Інтернет", „Web-дизайн та Web-програмування", „Технології проектування та адміністрування БД і СД ".

Мета лекції: розглянути особливості об'єктно-орієнтованого підходу до програмування; отримати базові знання, необхідні для розробки проектів мовою Java з використанням класів та об'єктів; формувати вміння аналізувати, узагальнювати, порівнювати, абстрагуватися, синтезувати знання, отримані при вивченні даної теми.

План лекції:

1. Принципи ООП
2. Класи і об'єкти
3. Пакети
4. Модифікатори доступу і інкапсуляція
5. Статичні члени і модифікатор static
6. Об'єкти як параметри методів
7. Внутрішні і вкладені класи
8. Спадкування
9. Абстрактні класи
10. Ієрархія наслідування і перетворення типів
11. Інтерфейси
12. Інтерфейси в механізмі зворотного виклику
13. Перерахування enum
14. Клас Object і його методи
15. Узагальнення (Generics)
16. Обмеження узагальнень
17. Спадкування і узагальнення
18. Довідкові типи і копіювання об'єктів

Опорні поняття: клас, об'єкт, шаблон, поле, метод, модифікатор, конструктори, ініціалізація, екземпляр класу, пакети, імпорт, інкапсуляція, спадкування, перевизначення, диспетчеризація, абстрактні класи, похідний клас, ієрархія наслідування, суперклас, інтерфейси, статичні методи, множинна реалізація, приватні методи, перерахування, узагальнення, обмеження, посилальні типи.

Інформаційні джерела:
Основна та допоміжна література:

1. Алгоритми і структура даних: Навчальний посібник / В.М.Ткачук. - Івано-Франківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. − 286 с.
2. Блинов И.Н., Романчик В. С. Java. Методы программирования : уч.-мет. пособие / И. Н. Блинов, В. С. Романчик. − Минск : издательство "Четыре четверти", 2013. − 896 с.
3. Васильев А. Н. Java. Объектно-ориентированное программирование: Учебное пособие. − СПб.: Питер, 2011. − 400 с.
4. Гамма, Э., Хелм, Р., Джонсон, Р., Влиссидес, Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. − СПб. : Питер, 2007. − 366 с.
5. Грязнова В. О., Єфіменко С. В. Основи методології програмування. - К.: ВПЦ "Київський університет", 2010.
6. Кингсли-Хьюджес Э., Кингсли-Хьюджес К. Справочник программиста. - М.: ООО "ИД Вильямс", 2007.
7. Хорстманн, К. С., Корнелл, Г. Библиотека профессионала. Java 2 : Том 1. Основы. − 8-е изд. − М. : Вильямс, 2013. − 816 с.
8. Шилдт Г. Полный справочник по Java SE 6. М.: Вильямс, 2010. − 1040 с.
9. Эккель Б. Философия Java. − 4-е изд. − СПб. : Питер, 2011. − 640 с.

Інтернет ресурси:
1. Java - Учебник для начинающих программистов : [Електрон. ресурс]. - Режим доступу: http://proglang.su/java
2. Программирование на Java: [Електрон. ресурс]. - Режим доступу: https://www.intuit.ru/studies/courses/16/16/info
3. Топ-5 полезных видеокурсов по Java : [Електрон. ресурс]. - Режим доступу: https://javarush.ru/groups/posts/528-top-5-poleznihkh-videokursov-po-java-chastjh-1--
4. Уроки по основам языка программирования JAVA для начинающих : [Електрон. ресурс]. - Режим доступу: https://www.fandroid.info/tutorial-po-osnovam-yazyka-programmirovaniya-java-dlya-nachinayushhih/
5. Энциклопедия языков программирования : [Електрон. ресурс]. - Режим доступу: http://progopedia.ru/
6. Язык программирования Java и среда NetBeans : [Електрон. ресурс]. - Режим доступу: https://www.intuit.ru/studies/courses/569/425/info

Навчальне обладнання, ТЗН, презентація тощо: ноутбук, проектор, мультимедійна презентація.

ВИКЛАД МАТЕРІАЛУ ЛЕКЦІЇ

Питання 1. Принципи ООП

ООП - це методологія програмування заснована на представленні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію спадкування.

Витоки ООП – програмування у великому

- Прості програми – це програми, що які задумуються, розробляються, супроводжуються і використовуються однією людиною – початківцем або професіоналом, що працює ізольовано.
- Складні програми - це промислові програмні продукти. Їм притаманна складність - єдиний розробник практично не в змозі охопити всі аспекти такої системи. Складність таких програм перевищує можливості людського інтелекту.

Складність викликається чотирма основними причинами:

- складністю реальної предметної області, з якої виходить замовлення на розробку;
- незадовільними способами опису поведінки великих дискретних систем.
- трудністю управління процесом розробки;
- гнучкістю програмних систем;

Боротьба зі складністю - декомпозиція

- Розділяй і володарюй (divide et impera) – це у сучасній інтерпретації принцип декомпозиції.
- Декомпозиція алгоритмічна – це виокремлення дій та встановлення їх порядку
- Декомпозиція об'єктна – спочатку виділяє абстрактні об’єкти з предметної області, а лише потім зосереджується на способі їх взаємодії.
- Однак ці способи взаємовиключають один одного – ми не можемо сконструювати складну систему одночасно двома способами.

Поняття об'єкта

- Будь-що є об'єктом. Будь предметна область розглядається як набір об'єктів
- Об'єкт - це конкретний упізнаваний предмет, одиниця або сутність (реальна чи абстрактна), що має чітко визначене функціональне призначення в даній предметній області

Приклади об'єктів

- Приклади матеріальних об'єктів: більярдна куля, тривимірний графічний примітив - куб, сфера і т.п.
- Приклади не матеріальних об'єктів, подій чи процесів: траєкторія руху кулі, хімічний процес, лінія перетину графічних примітивів

Поняття об'єкта в ООП

- Об'єкт - це абстракція предметної області, що має інтерфейс у вигляді іменованих операцій і власні дані, з обмеженням доступу до них
- Об'єкт має стан та поведінку

- Стан об'єкта характеризується переліком всіх властивостей даного об'єкту і поточними значеннями кожної з цих властивостей
- Властивість - це характеристики, що роблять даний об'єкт самим собою
- Приклад станів деякого кола:

Властивість	Значення
Центр	(10, 10)
Радіус	25

Поведінка об'єкта

- Стан об'єкта може змінитися тільки в результаті виклику методів
- Поведінка - це те, як об'єкт діє або реагує
- Поведінка - це набір операцій (методів) об'єкта
- Операція - це послуга, яку об'єкт може надати своїм клієнтам (іншим об'єктам)

Організація програми з точки зору ООП моделі

- Обчислення здійснюється шляхом взаємодії об'єктів між собою, при якому один об'єкт (клієнт) вимагає щоб інший об'єкт (сервер) виконав деяку дію
- Це відбувається шляхом відправки повідомлення клієнтом (відправником) серверу (отримувачу)

Поняття повідомлення

- Об'єкти взаємодіють відправляючи та отримуючи повідомлення
- Повідомлення – це запит на виконання деякої дії, що доповнюється аргументами необхідними для її виконання
- В якості реакції отримувач запускає деякий метод, щоб виконати наданий запит

Відмінність повідомлень від методів

- У повідомлень є конкретний отримувач
- Інтерпретація повідомлення залежить від отримувача та може різнитися у різних отримувачів
- Отримувач може по різному інтерпретувати (викликати різні методи) одне й те ж повідомлення в залежності від переданих аргументів

Поняття класу

- Клас - це деяка множина об'єктів, що мають спільну структуру та поведінку. Будь-який об'єкт є екземпляром класу
- Усі примірники одного класу будуть вести себе однаковим чином у відповідь на однакові запити
- Клас – шаблон множини об'єктів із спільною структурою і поведінкою

Принципи об'єктно-орієнтованого підходу

- Клас винен проектуватися і розроблятися з використанням принципів інкапсуляції, успадкування і поліморфізму
- Правильне застосування зазначених принципів підвищує абстрагування та покращує класифікацію предметної області, що в кінцевому рахунку дозволяти боротися зі складністю предметної області

Поняття абстрагування і класифікації

- Абстрагування (abstraction) - це уявлення предметної області (програми) у вигляді меншої кількості більш великих понять (блоків) і мінімізація зв'язків між ними
- Класифікація - це виділення істотних, стійких відносин (зв'язків) між поняттями (блоками)

Поняття інкапсуляції

- Інкапсуляція - це об'єднання даних з функціями їх обробки у поєднанні з приховуванням непотрібної для використання цих даних інформації
- Інкапсуляція - це приховування реалізації класу і відділення його внутрішнього представлення від зовнішнього (інтерфейсу)

Поняття інтерфейсу і реалізації

- Інтерфейс (interface) - це зовнішній вигляд класу, що виділяє його істотні риси і не показує внутрішній устрій
- Реалізація (implementation) - внутрішнє представлення класу, включаючи секрети його поведінки

Переваги використання інкапсуляції

- Підвищує ступінь абстракції програми – для написання програми не вимагається знання даних класу і реалізації його функцій
- Дозволяє змінити реалізацію класу без модифікації іншої частини програми, якщо інтерфейс залишився колишнім

Поняття спадкування

- Спадкування - це таке відношення між класами, коли один клас повторює структуру і поведінку іншого класу. Іншими словами, структура і поведінка передається від предка до нащадка

– Спадкування реалізує ставлення "is-a" між двома класами, тобто дочірній клас повинен бути частковим або спеціалізованим випадком батьківського класу

Переваги використання спадкування

– При спадкуванні загальні властивості і поведінка не описуються, що скорочує обсяг програми
– Виділення загальних рис різних класів в один клас-предок є потужним механізмом абстракції та класифікації

Поняття поліморфізму

– Поліморфізм - це використання одного імені для різних сутностей. При цьому різнорідні суті, виступаючи під одним ім'ям, сприймаються як однотипні
– Поліморфізм - це можливість обробки даних змінного типу
– Поліморфізм - це можливість оперувати об'єктами, не володіючи точним знанням їх типів

Переваги використання поліморфізму

– Дозволяє записувати алгоритми лише один раз і потім повторно їх використовувати для різних типів даних
– Звужує концептуальний простір, тобто зменшує кількість інформації, яку необхідно пам'ятати програмісту

Образне порівняння процедурного і об'єктно-орієнтованого підходів

– В об'єктно-орієнтованому підході в центрі уваги не "порядок виконання програми", а функціональні можливості об'єктів.
– "Порядок", в якому ці можливості будуть реалізовані в процесі виконання конкретної системи, є вторинною властивістю

Відмінності між процедурним і об'єктно-орієнтованим підходами

– У процедурному підході за допомогою покрокового уточнення вихідне завдання розбивається на все більш дрібні підзадачі, поки вони не стануть настільки простими, що їх можна буде реалізувати безпосередньо
– В об'єктно-орієнтованому підході спочатку виділяються класи, а лише потім визначаються їх методи. При цьому кожен метод пов'язаний з класом і клас відповідає за їх виконання
– У процедурному підході програма являє собою однорідну множину процедур, що обмінюються даними
– В об'єктно-орієнтованому підході класи надають зручний механізм кластеризації методів. Крім того приховують деталі представлення даних від будь-якого коду, крім своїх методів. Це означає, що якщо помилка програмування спотворює дані, то її легше знайти

– У процедурному підході неможливо отримати декілька копій одного модуля. Модуль - це набір пов'язаних даних і процедур, зібраних в окремому файлі. Модуль може мати інтерфейсну частину і реалізацію
– В об'єктно-орієнтованому підході на основі класу можна створити кілька об'єктів з однаковою поведінкою

Переваги об’єктно-орієнтованого підходу

– Об'єктна модель орієнтована на людське сприйняття світу: багато людей, які не мають поняття про те, як працює комп'ютер, знаходять цілком природним об'єктно-орієнтований підхід до систем
– Більш ефективна боротьба зі складністю програмного забезпечення
– Більш високий відсоток повторного використання коду
– Підвищення надійності програмного забезпечення
– Забезпечення можливості модифікації окремих компонентів програмного забезпечення без зміни інших його компонентів
– Підвищується рівень уніфікації розробки → придатність до повторного використання проектів
– Об'єктно-орієнтовані системи часто є більш компактними → економія часу і ресурсів → зменшення вартості
– Використання об'єктної моделі призводить до побудови систем на основі стабільних проміжних описів, що спрощує процес внесення змін. Це дає можливість розвиватися поступово і не призводить до повної її переробки навіть у разі істотних змін вихідних вимог
– Об'єктна модель зменшує ризик розробки складних систем, насамперед тому, що процес інтеграції розтягується на весь час розробки, а не перетворюється в одноразову подію
– Об'єктний підхід складається з ряду добре продуманих етапів проектування, що також зменшує ризики і підвищує впевненість у правильності прийнятих рішень

Висновок. Java є об'єктно-орієнтованою мовою. Це означає, що писати програми на Java потрібно із застосуванням об'єктно-орієнтованого стилю. І стиль цей заснований на використанні в програмі об'єктів і класів.

Об'єктно-орієнтоване програмування (ООП; іноді об'єктозорієнтоване програмування, об'єктно-орієнтоване програмування) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція. Одною з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих, коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Сьогодні багато мов програмування або підтримують ООП (PHP, Lua) або ж є цілком об'єктно-

орієнтованими (зокрема, Java, C#, C++, Python, Ruby і Objective-C, ActionScript 3, Swift, Vala).

Продовжити перегляд

Загальний висновок за темою лекції

1. Ідея класів – це основа об'єктно-орієнтованого програмування (ООП). Мета ООП – намагання зв'язати дані й функції для їх обробки в єдине ціле – клас. В класах об'єднуються структури даних і функції їх обробки. Ідея класів відображає будову об'єктів реального світу – оскільки кожний предмет або процес має свої властивості, будову, поведінку.
2. Клас – це визначений користувачем тип даних. В класі задаються властивості і поведінка якого-небудь, об'єкта у вигляді полів-даних і функцій для роботи з ними.
3. Парадигма програмування – це набір теорій, методів, стандартів, які використовуються при розробці та реалізації програм на комп'ютері. ООП часто називають новою парадигмою програмування, хоча її революційний поступ розпочався давно у минулому.
4. Інкапсуляція – об'єднання в єдине ціле даних і алгоритмів обробки цих даних. В ООП дані називаються полями, а алгоритми – методами або функціями-членами (methods, member functions).
5. Успадкування − властивість створення ієрархії класів, коли нащадки отримують від попередника поля і методи.
6. Поліморфізм (від гр. poly – багато і morphos форма, означає багато форм) – це властивість класів однієї ієрархії вирішувати схожі за змістом завдання за допомогою різних алгоритмів.
7. Абстрагування − один із головних засобів, що використовуються для розв'язання складних задач.
8. Абстракція − це досить суттєві характеристики деякого об'єкта, які відрізняють його від усіх інших видів об'єктів і, таким чином, чітко визначають особливості даного об'єкта з точки зору подальшого розгляду та аналізу.
9. Обмеження доступу - це процес захисту окремих елементів об'єкта, що не порушує суттєвих характеристик об'єкта як цілого.
10. Ієрархія – це аранжована та упорядкована система абстракцій.

Питання і завдання студентам для контролю знань

1. Що таке парадигма програмування?
2. Які парадигми програмування Ви знаєте?
3. Що таке об'єктно-орієнтоване програмування?
4. Які основні елементи об'єктної моделі Ви знаєте?
5. Що таке абстрагування?
6. Що таке інкапсуляція?
7. Що спільного та чим відрізняються абстракція і інкапсуляція?

8. Що таке модульність?
9. Що таке ієрархічність?
10. Які види ієрархічності Ви знаєте?
11. Що таке типізація?
12. Що таке паралелізм?
13. Що таке збережуваність?
14. Що таке інкапсуляція?
15. Наведіть визначення та опишіть зв’язок між поняттями класу, об’єкту та екземпляру класу.
16. Що таке спадкування (наслідування)?
17. Що таке поліморфізм?
18. У чому різниця вкладених і внутрішніх класів?
19. Як пов'язаний будь-який користувацький клас з класом Object?
20. Чим відрізняється абстрактний клас від інтерфейсу, в яких випадках його можна використовувати?
21. Чи можна отримати доступ до private змінних класу і якщо так, то яким чином?
22. Чи має сенс оголошувати метод private final?
23. Які особливості ініціалізації final змінних?
24. Що буде, якщо єдиний конструктор класу оголошений як final?
25. Які модифікатори за замовчуванням мають поля і методи інтерфейсів?
26. Чому не можна оголосити метод інтерфейсу з модифікатором final або static?
27. Які типи класів бувають в java (вкладені ... і.т.д.)?
28. Навіщо потрібні і які бувають блоки ініціалізації?
29. Який порядок виклику конструкторів і блоків ініціалізації двох класів: нащадка і його предка?
30. Де і для чого використовується модифікатор abstract?
31. Чи можна оголосити метод абстрактним і статичним одночасно?

Укладач: _____ Шевчук І.Б., доцент, д.е.н., доцент
(підпис) (ПІБ, посада, науковий ступінь, вчене звання)