ВСТУП ДО ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Тема 01
Об’єктно-орієнтоване програмування

План лекції

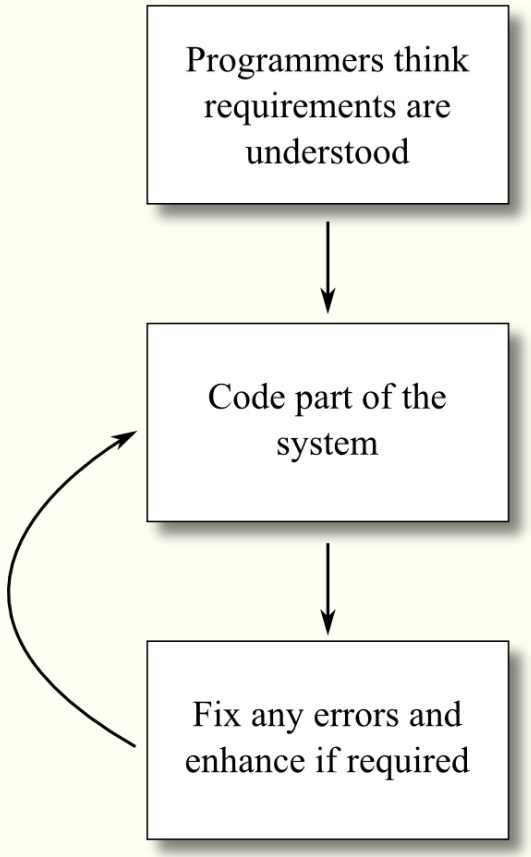
- **Базові концепції об’єктно-орієнтованого програмування.**
- **Огляд платформи .NET.**
- **Структура програми мовою C#**
- **Система типів мови програмування C#.**

Література до теми

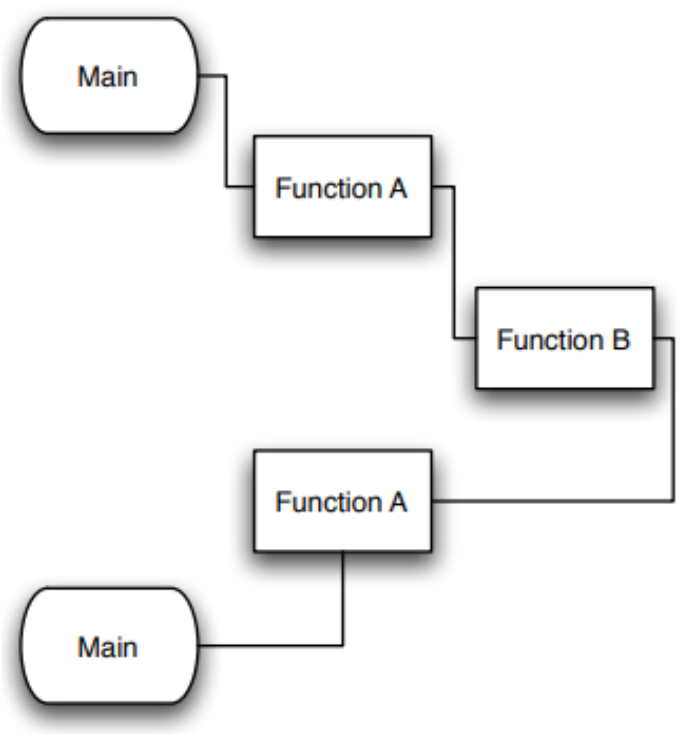
ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ: БАЗОВІ КОНЦЕПЦІЇ

Питання 1.1.
Об’єктно-орієнтоване програмування

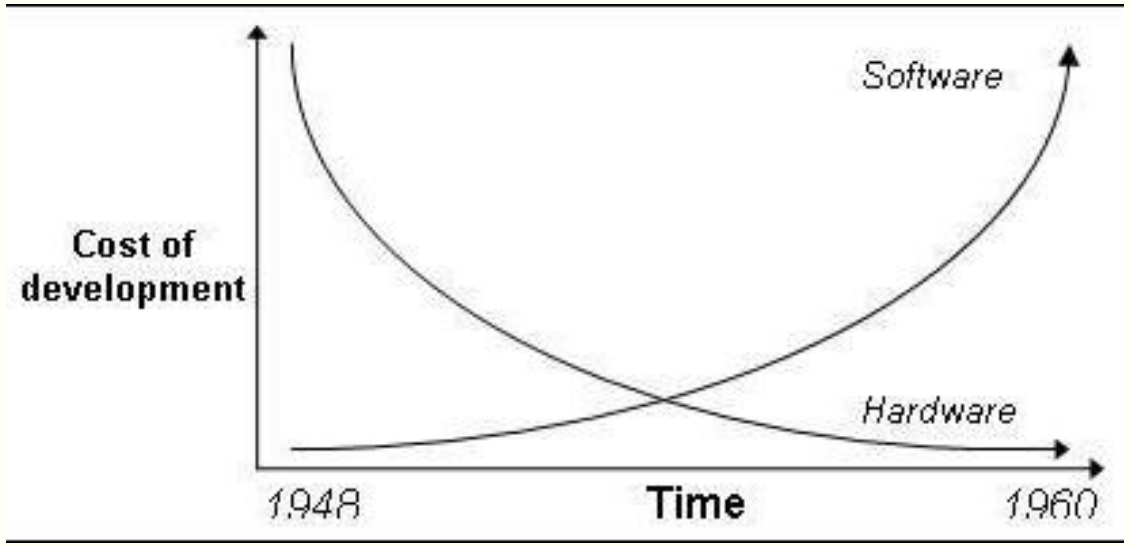
Криза програмного забезпечення (кінець 1960х – початок 1970х років)



- Наприкінці 1960х рр. проявилось багато складнощів при розробці ***крупних*** програмних систем.
- Стандартна у той час стратегія розробки – “code-and-fix”.
  - Не дозволяє оцінити часові та фінансові затрати на проект.
  - Не дозволяє спрогнозувати ризики та проектні недоліки ПЗ.
- Особливості процедурного (структурного) програмування:
  - Концентрація на алгоритмах (як робити?);
  - Великі програми розбиваються на функції;
  - Більшість функцій поділяють глобальні дані;
  - Дані відкрито переміщаються між функціями;
  - Функції перетворюють дані з однієї форми в іншу;
  - Застосовується підхід «згори-донизу» при проектуванні програм.



Криза програмного забезпечення (кінець 1960х – початок 1980х років)



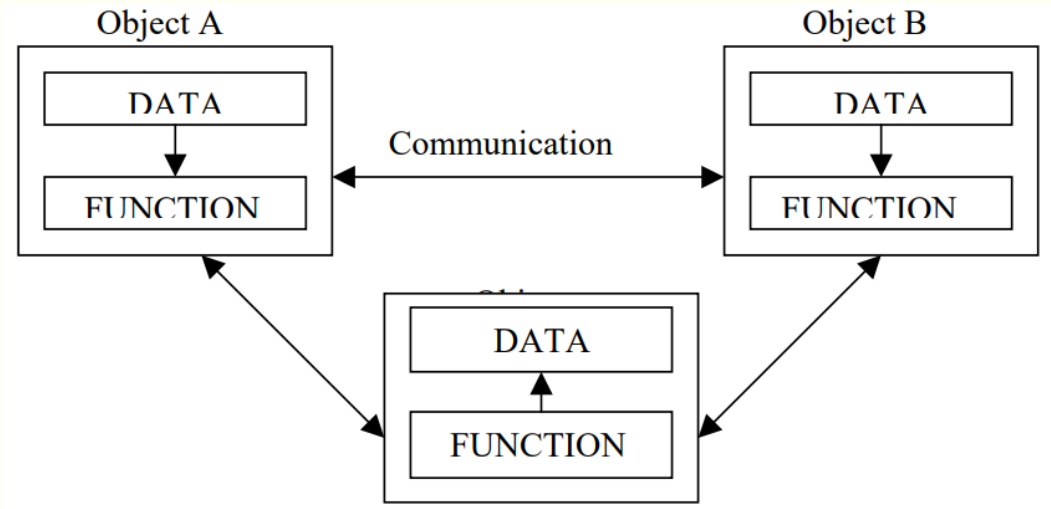
- Багато провалів у розробці програмних проектів спричинені нездатністю розширити маленькі програмні системи до більш складних та крупних систем. У результаті:
  - Перевищується бюджет
  - Горять дедлайни
  - Програмна система не виконує всі поставлені до неї вимоги
  - Система працює, проте нею дуже складно користуватись
  - Систему складно змінювати у відповідь на зміну вимог
  - Система не досягає стану завершеності
  - Розробники втрачають довіру замовника та користувачів ПЗ.

Питання, які виникли

- Як представити ***реальні сутності*** з предметної області при проектуванні програмних систем?
- Як забезпечити ***повторне використання*** частин коду та їх здатність до розширення?
- Як розробляти програмні модулі, ***підготовлені до майбутніх змін***?
- Як ***покращити продуктивність*** ПЗ та ***зменшити вартість*** його розробки?
- Як ***покращити якість*** ПЗ?
- Як ***планувати час*** розробки ПЗ?

Подолання кризи

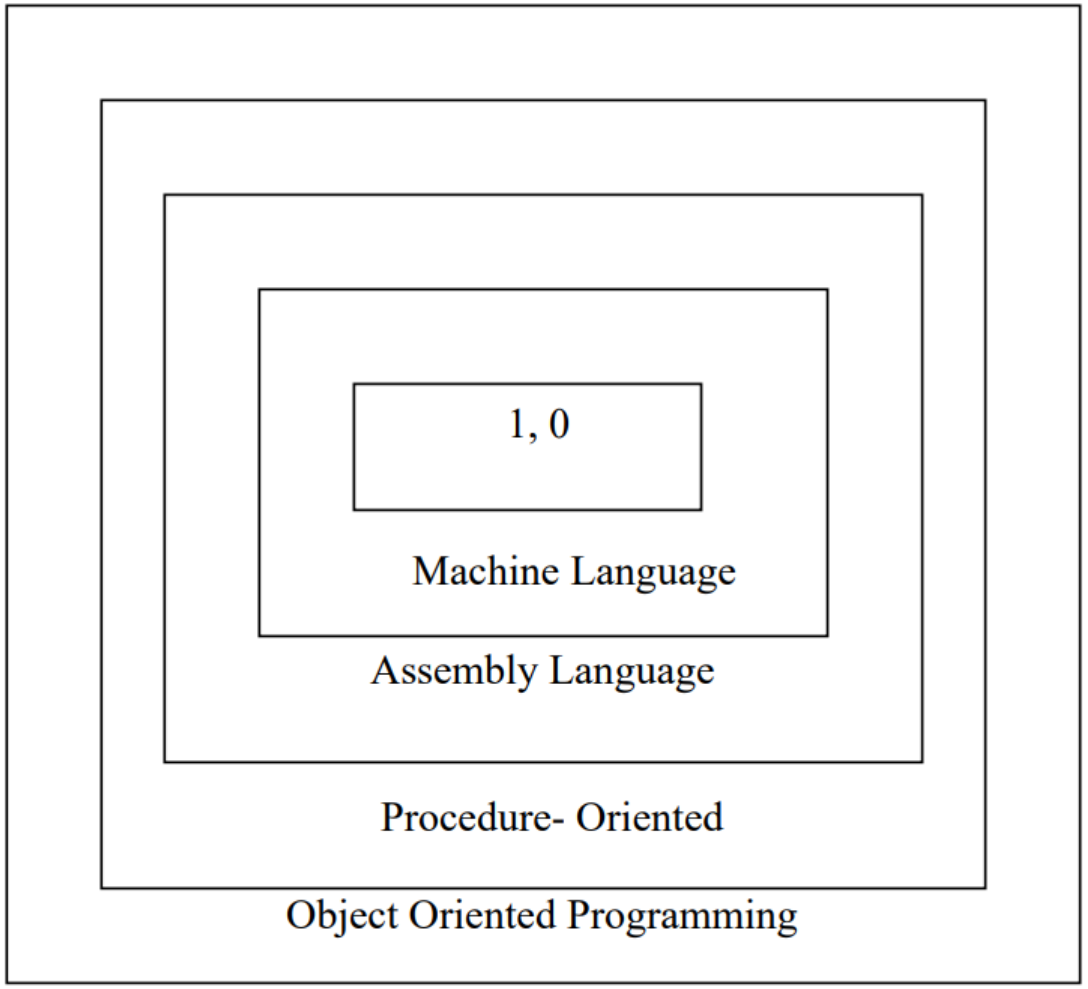
- Поява дисципліни «Програмна інженерія» (1968, конференція НАТО).
- Широкі дослідження «програмного забезпечення як технології»:
  - Розгляд історії мов програмування,
  - Проекти з сертифікації та підвищення кваліфікації програмістів
  - Стандартизація розробки ПЗ та формування груп стандартизації ПЗ (ACM, IEEE Computer Society)
- Спроби подолати недоліки процедурного програмування.
  - ООП розглядає ***дані як критичне ядро програми*** та ***не дозволяє їм вільно переміщуватись*** у програмній системі.
  - ООП намагається ближче ***поєднати дані з функціями***, які з цими даними працюють, та ***захистити дані від ненавмисних змін ззовні*** функції.
  - ООП дозволяє ***декомпозицію (розбиття) задачі на набір сутностей (об'єктів)*** та будувати дані з функціями навколо цих об'єктів.



Загальні характеристики об’єктно-орієнтованого програмування

- ***Концентрація на даних***, а не процедурах.
- Програми розбиваються на сутності, які називають ***об’єктами***.
- Структури даних проектуються так, щоб характеризувати об’єкти.
- ***Функції***, що оперують даними об’єкта, ***поєднуються в єдину структуру даних***.
- Дані приховуються (обмежується доступ до них) із зовнішніх функцій.
- ***Об’єкти можуть взаємодіяти*** один з одним ***за допомогою функцій***.
- Нові дані та функції можуть ***розширювати функціональність*** в потрібному місці.
- Застосовується ***підхід «знизу-вгору»*** про проектуванні програмних систем.

Концепції ООП

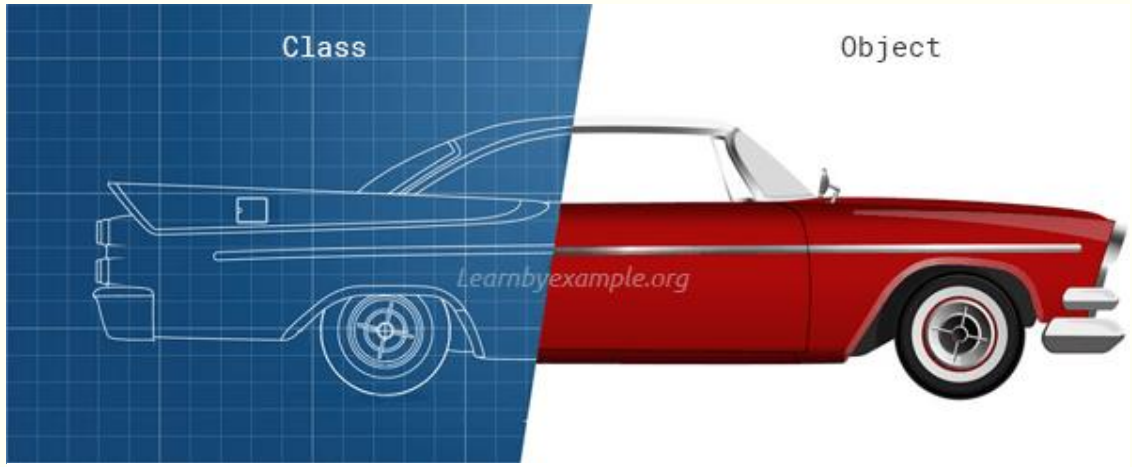


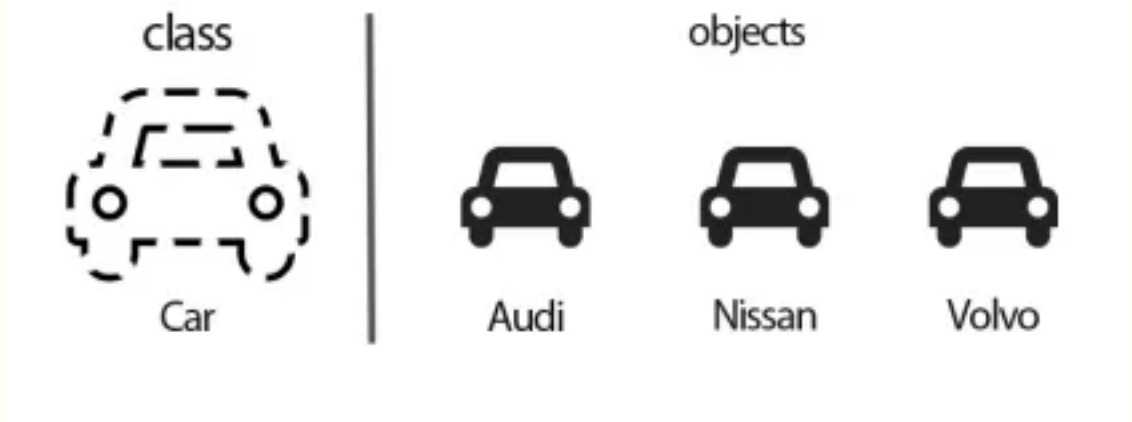
- Об’єктно-орієнтоване програмування доповнює структурне програмування кількома потужними концепціями:
  - об’єкти;
  - класи;
  - абстрагування даних та інкапсуляція;
  - наслідування;
  - поліморфізм;
  - динамічне зв’язування (Dynamic binding);
  - передача повідомлень (Message passing).

Об’єкти

- Об’єкти можуть представляти людину, місце, банківський рахунок, таблицю даних чи будь-яку іншу сутність, з якою працює програма.
  - Також можуть представляти визначені користувачем структури даних – вектори, час, списки тощо.
  - Об’єкти обираються так, щоб максимально відповідати об’єктам реального світу.
  - Об’єкти займають місце в пам’яті та мають асоційовану з ними адресу.
- Коли програма виконується, об’єкти взаємодіють між собою шляхом ***передачі повідомлень***.
  - Для об’єктів «клієнт1» та «рахунок1», наприклад, від клієнта надсилається повідомлення щодо отримання балансу на рахунку.
  - Об’єкти можуть взаємодіяти, не знаючи деталей організації даних чи коду один одного.
- Характеристики об’єкта:
  - ***Стан (state)***: представляє дані об’єкта;
  - ***Поведінка (behavior)***: описує доступні операції з об’єктом;
  - ***Ідентичність (identity)***: використовується компілятором для унікальної ідентифікації об’єктів.

Класи



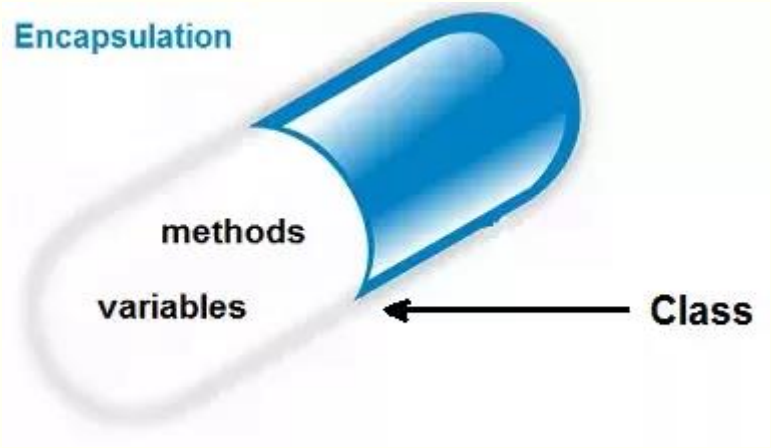


- ***Клас*** – це користувацький тип, який описує структуру об'єкта.
  - Фактично, Об'єкт – це змінна типу Клас.
  - Описавши клас, зазвичай можна створити довільну кількість ***об'єктів (екземплярів класу, instances)***.
- Об'єкт є екземпляром класу, а клас є колекцією об'єктів (даних).

Student	Circle	SoccerPlayer	Car
name gpa	radius color	name number xLocation yLocation	plateNumber xLocation yLocation speed
getName() setGpa()	getRadius() getArea()	run() jump() kickBall()	move() park() accelerate()

Examples of classes

Абстрагування даних та інкапсуляція

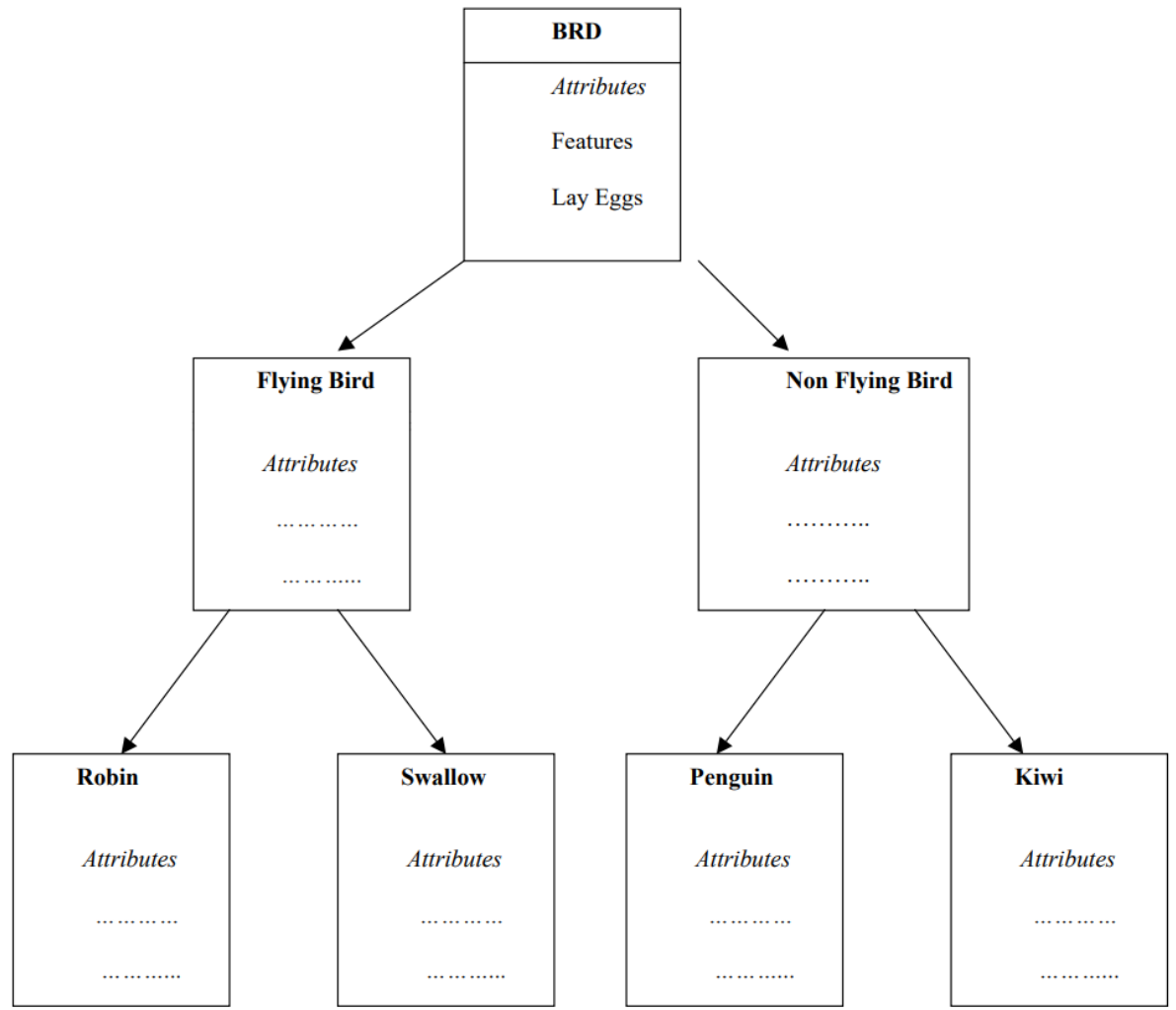


Student
name gpa
getName() setGpa()

Circle
radius color
getRadius() getArea()

- Поєднання даних та функцій у єдиний модуль – клас – називають ***інкапсуляцією***.
  - Зазвичай доступ до даних мають тільки функції, які обгортаються в єдиний клас з цими даними.
  - Ці функції надають ***інтерфейс взаємодії між об'єктом та програмою***.
  - Заборону прямого доступу до даних об'єкта називають ***приховуванням даних (data hiding)*** або ***приховуванням інформації (information hiding).***
- Під ***абстракцією*** розуміють представлення основних характеристик без урахування чи пояснення деталей їх реалізації.
  - Класи інкапсулюють всі основні властивості сконструйованих з них об'єктів, записуючи абстрактні ***атрибути (поля класу, дані-члени, data members).***
  - Функції, що оперують атрибутами класу, називають ***методами (функціями-членами класу, member function).***

Наслідування



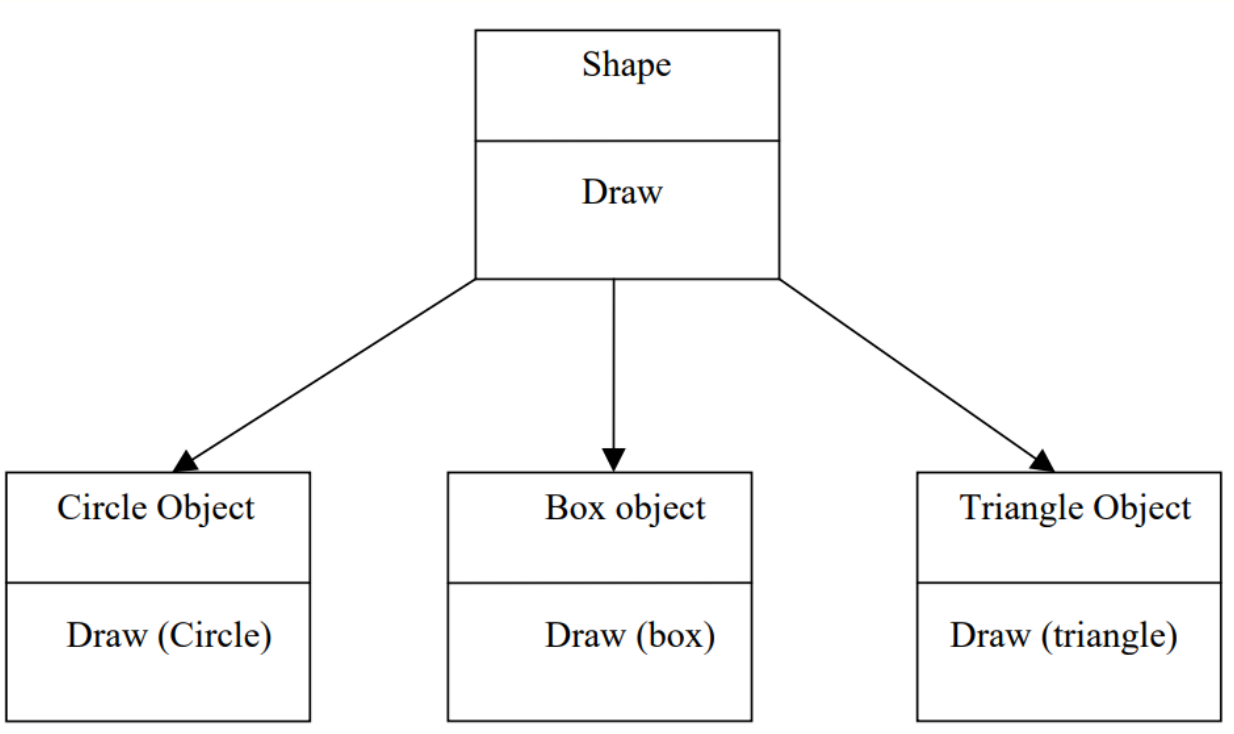
Процес, у ході якого один клас набуває властивостей іншого класу.

- Породжує ***ієрархічну класифікацію*** при проектуванні.

В ООП концепція наслідування реалізує ідею ***повторного використання коду (reusability)***.

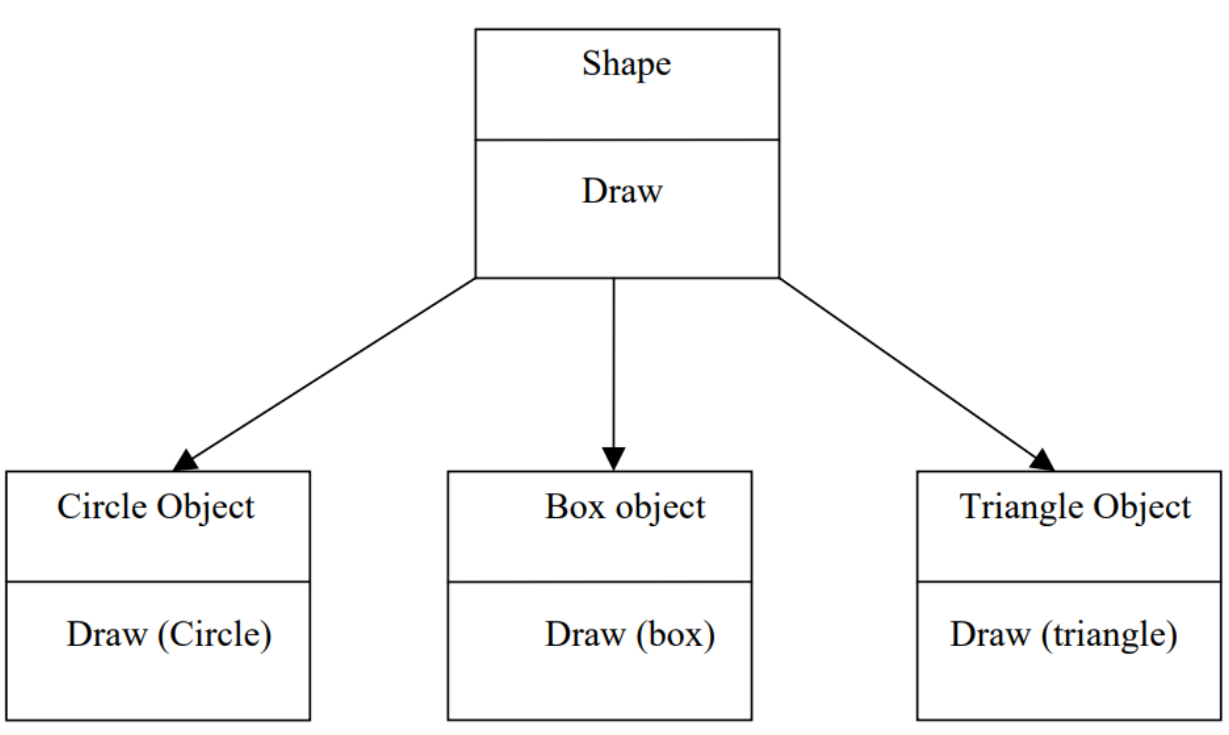
- Існує можливість додавати нові можливості до існуючих класів, не змінюючи їх, шляхом ***породження (derivation) дочірнього класу***.
- Наслідування дозволяє повторно використовувати раніше створені класи зі схожою функціональністю, відбираючи з них тільки потрібні можливості.

Поліморфізм



- Здатність поведінки проявлятись по-різному для різних екземплярів класів.
  - Поведінка залежить від типу даних, які використовуються.
  - Використання однієї назви методу для виконання різних типів завдань називають перевантаженням методу (function overloading, method overloading).
  - Використання одної форми запису оператора для виконання ним різних типів завдань називають перевантаженням операторів (operator overloading).
  - Поліморфізм широко використовується в реалізації наслідування.

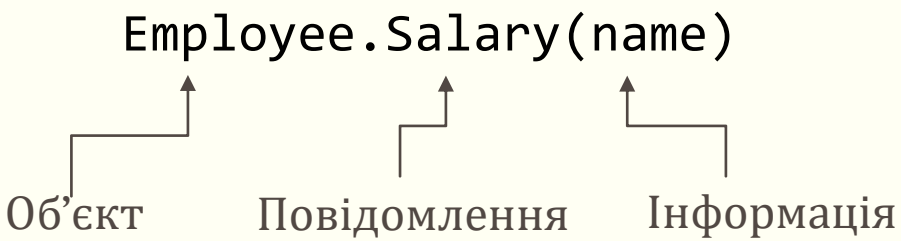
Динамічне зв’язування



- Передбачає, що код, пов’язаний із процедурним викликом, не розглядається до моменту самого виклику під час виконання.
  - Динамічний тип компілятор перевіряє тільки під час виконання коду, але не в ході компіляції.
- Динамічне зв’язування пов’язане з поліморфізмом та наслідуванням.
  - Поліморфний виклик методу, залежить від динамічного типу об’єкта (посилання), якому належить цей метод.
  - У прикладі об’єкти кожного з класів мають метод Draw(), які мають різні реалізації залежно від класу.
  - Під час виконання коду відбувається співставлення типу об’єкта та виклик відповідного методу.

Передача повідомлень

- Процес програмування ОО мовою включає наступні кроки:
  - Створення класів, які визначають об’єкт та його поведінку
  - Конструювання об’єктів з оголошення класу (class definition)
  - Встановлення взаємодії між об’єктами.
- Об’єкти взаємодіють один з одним, надсилаючи та отримуючи інформацію (подібно до людей).
  - Повідомлення – це запит до об’єкта на виконання його методу.
  - Передача повідомлень включає назву об’єкта, назву методу (повідомлення) та передану інформацію (аргументи методу).
  - Взаємодія з об’єктом можлива, поки він не знищений (destroyed).



Основні переваги ООП

- За допомогою наслідування можна усунути надмірний код, використовуючи вже існуючі класи.
- Програму (в теорії) можна побудувати зі стандартних робочих модулів, які взаємодіють один з одним, що зберігає час та підвищує продуктивність.
- Приховування даних допомагає будувати більш безпечні програми, в які не зможе втручатись зовнішній код.
- Існує можливість відображення об’єкта з предметної області в програму.
- Простіше розбити командну роботу над проектом.
- ОО системи набагато простіше масштабуються від малих до великих.
- Спрощується управління складністю ПЗ.

Галузі застосування ООП

- Програмне забезпечення з графічним інтерфейсом користувача (вікна, кнопки, меню та ін.)
- Клієнт-серверні системи.
- Hypertext, Hypermedia.
- Бази даних .
- Імітування та моделювання.
- Штучний інтелект та експертні системи
- Нейронні мережі та паралельне програмування
- Системи автоматизації рішень та робочих місць
- CAD/CAM системи.
- Системи реального часу (Real Time Systems)

ДЯКУЮ ЗА УВАГУ!

Наступне питання: Огляд платформи .NET