Bài 20
SOLID & OOAD

Module: ADVANCED PROGRAMMING WITH JAVA

Mục tiêu

- Trình bày được ý nghĩa của nguyên lý SOLID
- Trình bày được ý nghĩa của OOAD

Program to an interface, not an concrete implementation

S.O.L.I.D

- SOLID là viết tắt của các từ:
  - **S** - Single-responsiblityprinciple– Nguyên lý Trách nhiệm Duy nhất
  - **O** - Open-closed principle – Nguyên lý đóng mở
  - **L** - Liskov substitution principle – Nguyên lý Thay thế Liskov
  - **I** - Interface segregation principle– Nguyên lý Phân tách Interface
  - **D** - Dependency Inversion Principle– Nguyên lý Đảo ngược Phụ thuộc
- SOLID bao gồm các nguyên lý quan trọng bậc nhất cần tuân thủ khi thiết kế kiến trúc phần mềm nhằm đạt được mục đích:
  - Dễ mở rộng
  - Dễ bảo trì

Single-responsiblity principle

- Nguyên lý Trách nhiệm Duy nhất
- Mỗi lớp chỉ nên đảm nhiệm một nhiệm vụ duy nhất
- Lí do:
  - Dễ quản lý mã nguồn
  - Các lớp tập trung vào nhiệm vụ của mình
  - Giảm tính phụ thuộc giữa các thành phần
  - Có thể phát triển đồng thời các lớp độc lập với nhau
  - Dễ dàng mở rộng
  - Dễ dàng bảo trì

Open-closed principle

- Nguyên lý Đóng-mở
- Các đối tượng (hoặc thực thể) nên mở đối với việc mở rộng, nhưng đóng đối với việc thay đổi
- Nghĩa là: Có thể dễ dàng mở rộng một lớp mà không cần thay đổi mã nguồn của lớp đó
- Lí do:
  - Dễ mở rộng
  - Dễ thay đổi

Liskov substitution principle

- Nguyên lý Thay thế Liskov
- Các lớp con có thể được sử dụng thay thế cho các lớp cha
- Nghĩa là: Nếu S là một lớp con của T, thì các đối tượng của lớp T có thể được thay thế bằng các đối tượng của lớp S mà không làm ảnh hưởng tới bất cứ hành vi nào của hệ thống
- Lí do:
  - Tránh các sai sót khi mở rộng thiết kế

Interface segregation principle

- Nguyên lý Phântách Interface
- Không nên bắt buộc phải triển khai một Interface nếu không cần đến nó, cũng không nên bắt buộc phải phụ thuộc vào các phương thức mà không cần đến chúng
- Nghĩa là: Các Interface chỉ nên chứa các phương thức cần thiết vừa đủ với mục đích của nó. Nên tách các Interface thành nhiều Interface nếu các phương thức của chúng không liên quan chặt chẽ đến nhau.

Dependency Inversion Principle

- Nguyên lý Đảo ngược Phụ thuộc
- Trừu tượng (abstraction) không nên phụ thuộc vào chi tiết. Chi tiết nên phụ thuộc vào trừu tượng.
- Các module ở mức trên không nên phụ thuộc vào các module ở mức dưới, mà cả hai nên phụ thuộc vào Trừu tượng (abstraction)

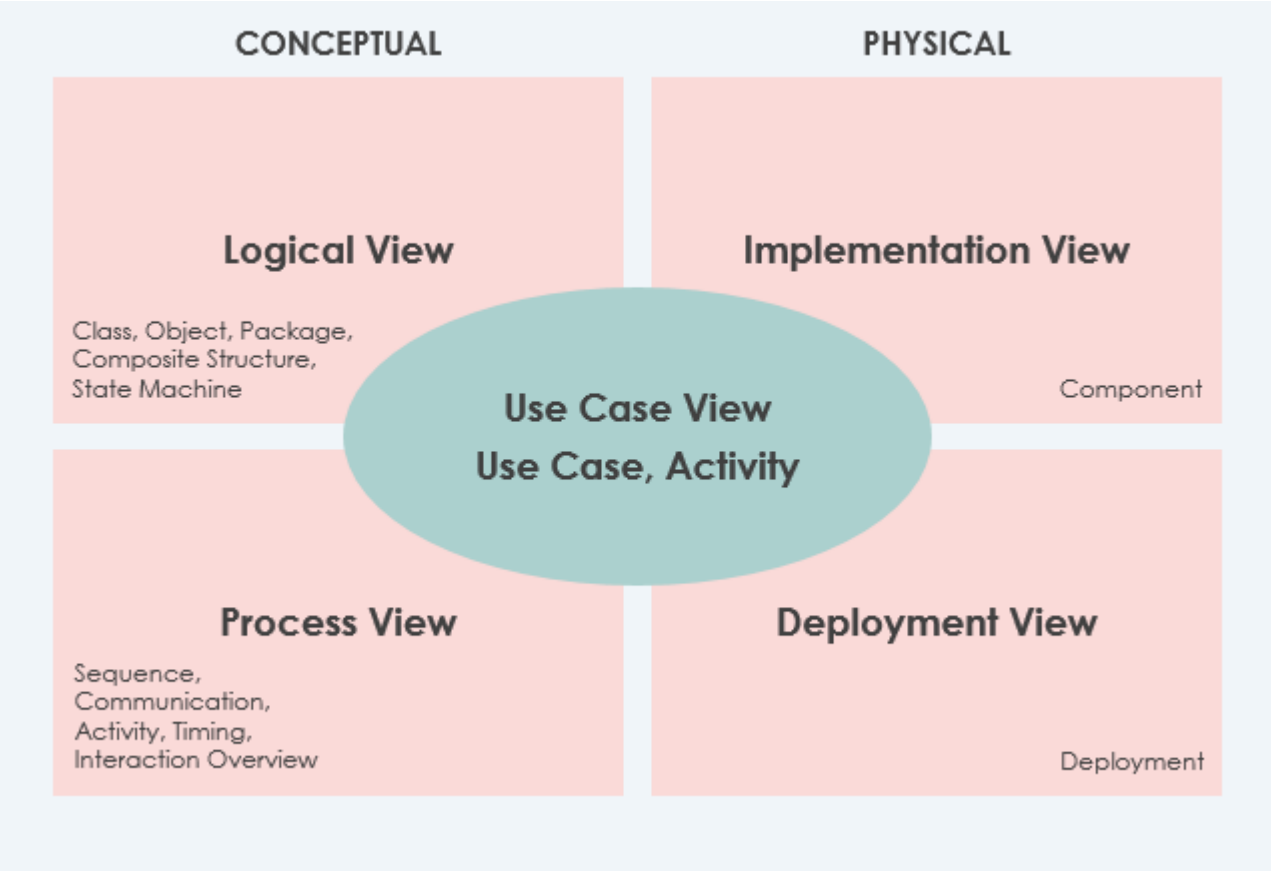
OOAD

- OOAD là viết tắt của Object Oriented Analysis and Design
- Để phân tích và thiết kế một phần mềm thì có nhiều cách làm, một trong những cách làm đó là xem hệ thống gồm những đối tượng sống trong đó và tương tác với nhau. Việc mô tả được tất cả các đối tượng và sự tương tác của chúng sẽ giúp chúng ta hiểu rõ hệ thống và cài đặt được nó. Phương thức này gọi là Phân tích thiết kế hướng đối tượng (OOAD)

UML

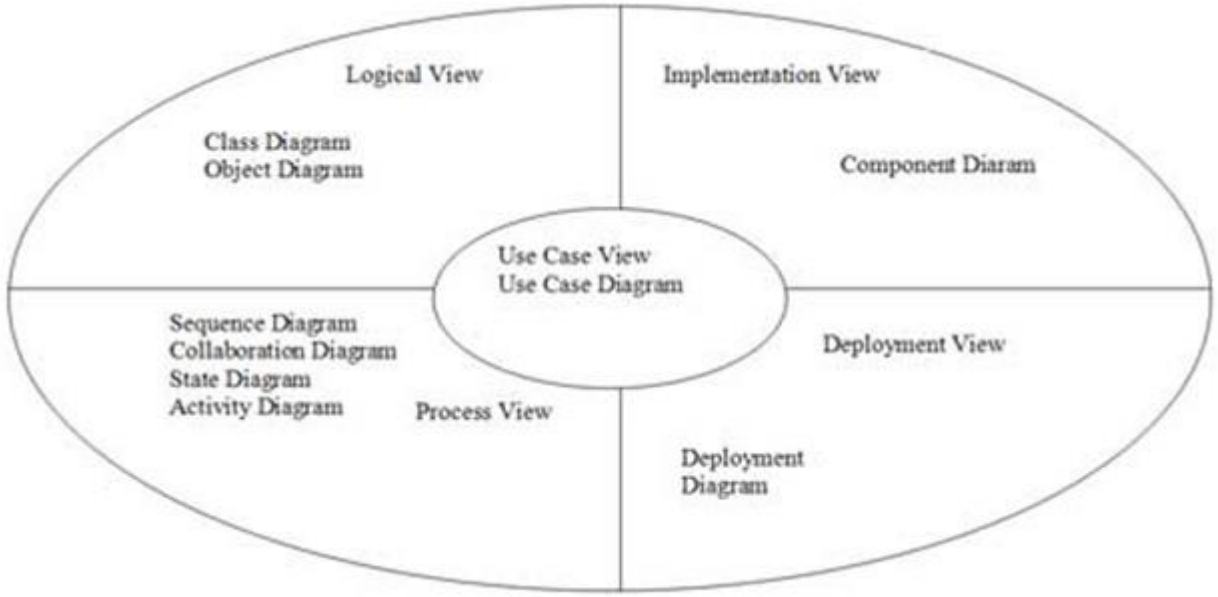
- UML là ngôn ngữ mô hình hóa hợp nhất dùng để biểu diễn hệ thống. Nói một cách đơn giản là nó dùng để tạo ra các bản vẽ nhằm mô tả thiết kế hệ thống.
- OOAD sử dụng UML bao gồm các thành phần sau:
  - View (góc nhìn)
  - Diagram (bản vẽ)
  - Notations (ký hiệu)
  - Mechanisms (quy tắc)

View



Các View trong OOAD sử dụng UML

Diagram



Các bản vẽ trong OOAD

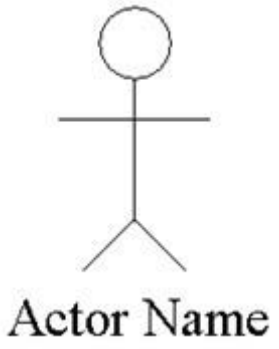
Notations

Một vài ký hiệu:

Use Case

Class

Actor



Mechanisms

Mechanisms là các qui tắc để lập nên bản vẽ, mỗi bản vẽ có qui tắc riêng và bạn phải nắm được để tạo nên các bản vẽ thiết kế đúng. Các qui tắc này chúng ta sẽ bàn kỹ trong các bài về các bản vẽ.

Tổng kết

- SOLID bao gồm các nguyên lý quan trọng bậc nhất cần tuân thủ khi thiết kế kiến trúc phần mềm nhằm đạt được mục đích: Dễ mở rộng và Dễ bảo trì
- Sử dụng UML để phân tích và thiết kế hướng đối tượng.

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: Design Pattern