Đặc trưng của OOP

Tính kế thừa

- **Định nghĩa:** Thừa hưởng các thuộc tính, hành vi từ một lớp khác.
- **Biểu hiện:** Class có thể kế thừa (extends) lớp cha và triển khai (implements) các interface.
- **Lợi ích:**
  - Giúp tái sử dụng code đã có nhưng vẫn duy trì một hệ thống phân cấp thống nhất.
  - Lớp cha đơn giản và tổng quát hơn lớp con. Lớp con cụ thể và đa dạng hơn lớp cha.
- **Khác:** C# không hỗ trợ đa kế thừa nhưng có hỗ trợ sử dụng Interface. Trong một số trường hợp có thể sử dụng interface để mô phỏng đa kế thừa.
- **Ví dụ:** Lớp cha là lớp động vật có các phương thức di chuyển, kêu, ăn… các lớp con như chó, mèo, chim, cá… kế thừa lớp cha động vật thì đều có các hành vi di chuyển, kêu, ăn…

Tính đóng gói

- **Định nghĩa:** Giữ tất cả các thông tin quan trọng (data, method) bên trong một lớp, chỉ cho phép truy cập nhưng thông tin cần thiết.
- **Biểu hiện:** Xác định mức độ truy cập thông qua các accsess modifiers public, internal, protected, private
- **Lợi ích:**
  - Tăng cường bảo mật, bảo vệ dữ liệu quan trọng khỏi truy cập trái phép.
  - Hạn chế truy cập thông tin không cần giúp dễ đọc, dễ hiểu, dễ bảo trì hơn.
  - Cho phép thay đổi cách hoạt động bên trong lớp mà không ảnh hưởng đến chương trình.
  - Tăng khả năng tái sử dụng của class trong các dự án khác nhau.
- **Ví dụ:** Lớp chó có thuộc tính trạng thái (đói, no) là private để thay đổi trạng thái này người dùng phải sử dụng method public cho ăn (feed) không thể truy cập trực tiếp vào thuộc tính trạng thái.

Tính trừu tượng

- **Định nghĩa:** chỉ tiết lộ những thông tin cần thiết của một đối tượng. Bỏ qua những chi tiết phức tạp bên trong.
- **Biểu hiện:** Thể hiện thông qua abstact class và interface. Thưởng chỉ đưa ra nhiệm vụ mà không nói cụ thể cách triển khai. Cách triển khai được thực hiện ở các class kế thừa nó.

- **Lợi ích:**
  - Đơn giản hoá, người dùng chỉ cần quan tâm đến các thuộc tính, phương thức được công khai không cần quan tâm đến cách thức hoạt động của nó.
  - Dễ dàng thay đổi cách thức hoạt động bên trong mà không ảnh hưởng đến phần còn lại của chương trình.
  - Tăng khả năng tái sử dụng
- **Ví dụ:**
  - Khi cho động vật ăn thì nó thực hiện việc ăn. Người nuôi không cần quan tâm đến bộ não con chó ra lệnh gì, con chó nhai như thế nào...
  - Lái một chiếc ô tô thì người lái chỉ cần quan tâm đến côn, ga, số, phanh, vô lăng không cần quan tâm đến cách động cơ đốt nhiên liệu như nào các bánh răng truyền động ra sao để chuyển hướng...

Tính đa hình

- **Định nghĩa:** Cho phép các đối tượng thuộc các lớp khác nhau có thể phản hồi một message theo nhiều cách khác nhau.
- **Biểu hiện:** Overiding method, overloading method
  - Overiding method (trong runtime): lớp con phản hồi khác lớp cha của nó.
  - Overloading method (compile time): Phương thức có cùng tên nhưng khác số lượng tham số và có thể khác dữ liệu trả về
- **Lợi ích:**
  - Giảm thiểu lặp code, giúp dễ bảo trì hơn.
  - Cho phép chương trình thích ứng với các trường hợp khác nhau mà không cần sửa code nhiều.
  - Giúp tạo ra các mô hình dễ đọc, dễ hiểu.
- **Ví dụ:** Các động vật kế thừa lớp động vật đều có hành động di chuyển, kêu nhưng chó, chim, cá thực hiện các hành động di chuyển, kêu hoàn toàn khác nhau.

Revision #1
Created 2025-10-12 18:28:41 UTC by ThanhDV
Updated 2025-10-12 18:29:39 UTC by ThanhDV