**1. Hướng đối tượng là gì?**

- Hướng đối tượng là kĩ thuật mô hình hóa một hệ thống thế giới thực trong phần mềm dựa trên các đối tượng.

**2. Đối tượng là gì?**

- Đối tượng (object) là khái niệm trung tâm của OOP, nó là một mô hình của một thực thể hay khái niệm trong thế giới thực. Việc mô hình hóa này bao gồm xác định các đối tượng tham gia bài toán – những cái làm nhiệm vụ gì đó hoặc bị làm gì đó.

**3. Lập trình theo kiểu hướng đối tượng là gì?**

- Lập trình theo kiểu hướng đối tượng là hoạt động định nghĩa các thể loại của các đối tượng đó ở hình thức các khuôn mẫu để tạo ra chúng.

**4. Lớp và đối tượng? Quan hệ giữa chúng?**

- Lớp đối tượng (class) là khuôn mẫu để sinh ra đối tượng

- Đối tượng là thể hiện (instance) của một lớp. Đối tượng có thuộc tính (dữ liệu), hành vi (phương thức)

- Đối tượng bắt đầu tồn tại khi hệ thống gọi một hàm khởi tạo (constructor) để khởi tạo giá trị cho các thuộc tính của đối tượng.

- Đối tượng được thao tác thông qua tham chiếu. Tham chiếu đóng vai trò gần giống như một con trỏ . Đối tượng phải được tạo ra một cách tường minh bằng toán tử new

Quan hệ giữa lớp và đối tượng gần giống như quan hệ giữa kiểu dữ liệu và các biến thuộc kiểu dữ liệu đó. Các đối tượng được tạo ra khi chương trình chạy, và lớp là khuôn mẫu mà từ đó có thể tạo ra các đối tượng thuộc lớp đó. Mỗi đối tượng được tạo ra từ một lớp được gọi là một thực thể (instance) của lớp đó. Một chương trình khi được viết là sự kết hợp của các lớp khác nhau. Còn khi chạy, nó là một tập hợp các đối tượng hoạt động và tương tác với nhau, các đối tượng này được sinh ra từ các lớp cấu thành nên chương trình đó.

**5. Phương thức trùng tên (overload)**

- Có thể định nghĩa các phương thức trùng tên, tuy nhiên phải phân biệt bởi danh sách tham số

6. Phương thức khởi tạo

- Phương thức khởi tạo là phương thức đặc biệt được gọi tự động ngay sau khi đối tượng được tạo ra nhằm mục đích chính là khởi tạo cho các thuộc tính của đối tượng

- Có tên trùng với tên lớp

- Không nhận giá trị trả lại

- Mỗi khi đối tượng được tạo ra bởi toán tử new, hệ thống sẽ tự động gọi phương thức khởi tạo. nếu không khai báo, hệ thống sẽ gọi constructor mặc định là một phương thức rỗng

7. Kiểu dữ liệu nguyên thủy trong java:

Java cung cấp các kiểu nguyên thủy

- số: byte, short, int, long, float, double (không có khái niệm unsigned, kích thước cố định trên mọi platform)

- logic: boolean

- ký tự: char

- Dữ liệu kiểu nguyên thủy không phải là đối tượng

- Tồn tại lớp đối tượng tương ứng(Wrapper class): Integer, Float,..

8. Tham chiếu

Đối tượng được thao tác thông qua tham chiếu

- là con trỏ tới đối tượng

- thao tác trực tiếp tới thuộc tính và phương thức

- không có các toán tử con trỏ

- phép gán (=) không phải là phép toán copy nội dung đối tượng

- tham chiếu được lưu trữ trong vùng nhớ static/stack như các con trỏ trong C/C++

9. Toán tử New

- Phải tạo mọi đối tượng một cách tường minh bằng toán tử new

- cấp phát vùng nhớ động

- được tạo trong bộ nhớ Heap

10. Phép gán “=”

- Phép gán không phải là copy thông thường

- copy nội dung của tham chiếu

- 2 tham chiếu sẽ tham chiếu đến cùng đối tượng

11. Toán tử quan hệ “==”

- So sánh nội dung của các dữ liệu kiểu nguyên thủy (int, long, float, …)

- So sánh nội dung của tham chiếu chứ không so sánh nội dung của đối tượng do tham chiếu trỏ đến

12. Tham chiếu this

Java cung cấp tham chiếu this để trỏ tới chính đối tượng đang hoạt động

- this được sử dụng vào các mục đích như:

- tham chiếu tường minh đến thuộc tính và phương thức của đối tượng

- truyền tham số và trả lại giá trị

- dùng để gọi constructor

13. Phương thức và thuộc tính static

- Có thể khai báo phương thức và thuộc tính là tĩnh (static)

- Độc lập với đối tượng( có thể sử dụng mà không cần có đối tượng)

- Khai báo dữ liệu dùng chung

Phương thức tĩnh: không sử dụng được thuộc tính thông thường (non-static), không gọi được các phương thức thông thường

14. Tính kế thừa

- Dựa trên quan hệ is\_a

- Cho phép xây dựng một lớp mới (lớp Con), kế thừa và tái sử dụng các thuộc tính, phương thức dựa trên lớp cũ (lớp Cha) đã có trước đó

- Các lớp Con kế thừa toàn bộ thành phần của lớp Cha và không cần phải định nghĩa lại. Lớp Con có thể mở rộng các thành phần kế thừa hoặc bổ sung những thành phần mới.

+ Thêm các thuộc tính mới

+ Thêm các phương thức mới

+ Hiệu chỉnh các phương thức đã có

- Lớp con không thừa kế các biến / phương thức private của lớp cha.

- Thuật ngữ:

+ Lớp cơ sở, lớp cha

+ Lớp dẫn xuất, lớp con

15.Từ khóa final

- Thuộc tính final: hằng số, chỉ được gán giá trị khởi tạo một lần, không thay đổi được giá trị

- Phương thức final: không cho phép định nghĩa lại ở lớp dẫn xuất

- Tham số final: không thay đổi được giá trị của tham chiếu

- Lớp final: không định nghĩa được lớp dẫn xuất

16. Tính đa hình

Theo nghĩa tổng quát, đa hình là khả năng tồn tại ở nhiều hình thức. Trong hướng đối tượng, đa hình đi kèm với quan hệ thừa kế và có hai đặc điểm sau:

(1) các đối tượng thuộc các lớp dẫn xuất khác nhau có thể được đối xử như nhau, như thể chúng là các đối tượng thuộc lớp cơ sở, chẳng hạn có thể gửi cùng một thông điệp tới đối tượng;

(2) khi nhận được cùng một thông điệp đó, các đối tượng thuộc các lớp dẫn xuất khác nhau hiểu nó theo những cách khác nhau.

Là một trong ba đặc điểm quan trọng của lập trình hướng đối tượng

Cho phép nhìn nhận đối tượng theo các cách khác nhau, giải nghĩa thông điệp theo các cách thức khác nhau

Dựa trên cơ chế chuyển kiểu và liên kết động

Tạo sự mềm dẻo và tính mở cho thiết kế phần mềm

Tăng khả năng sử dụng lại

17. Tính đóng gói

Tính đóng gói cho phép che giấu thông tin và những tính chất xử lý bên trong của đối tượng. Các đối tượng khác không thể tác động trực tiếp đến dữ liệu bên trong và làm thay đổi trạng thái của đối tượng mà bắt buộc phải thông qua các phương thức công khai do đối tượng đó cung cấp.

Tính chất này giúp tăng tính bảo mật cho đối tượng và tránh tình trạng dữ liệu bị hư hỏng ngoài ý muốn.

- private: chỉ trong cùng 1 lớp mới có thể truy cập các biến/phương thức private

- default: các biến/phương thức với mức truy nhập mặc định của một lớp chỉ có thể được truy nhập bởi mã nằm bên trong cùng một gói với lớp đó.

- protected: các biến/phương thức với mức protected của một lớp chỉ có thể được thừa kế bởi các lớp con cháu của lớp đó, kể cả nếu lớp con đó không nằm trong cùng một gói với lớp cha.

- public: mã ở bất cứ đâu cũng có thể truy nhập các thứ public (lớp, biến thực thể, biến lớp, phương thức, hàm khởi tạo...)

18. Tính trừu tượng

Tính trừu tượng giúp loại bỏ những thứ phức tạp, không cần thiết của đối tượng và chỉ tập trung vào những gì cốt lõi, quan trọng.

Một lớp cha không bao giờ được dùng để tạo đối tượng được gọi là lớp cơ sở trừu tượng, hay ngắn gọn là lớp trừu tượng (abstract class). Với những lớp thuộc diện này, trình biên dịch sẽ báo lỗi bất cứ đoạn mã nào định tạo thực thể của lớp đó.

- Lớp trừu tượng: tạo ra các lớp cơ sở để tái sử dụng mà không muốn tạo ra đối tượng thực của lớp

- Để thống nhất giao diện, có thể khai báo các phương thức tại lớp cơ sở nhưng được cài đặt thực tế tại lớp dẫn xuất. Các lớp dẫn xuất khác nhau có cách cài đặt khác nhau

- Phương thức trừu tượng: Là phương thức bắt buộc phải định nghĩa lại (chuyên biệt hóa) tại lớp dẫn xuất

• Một lớp trừu tượng có thể có các phương thức trừu tượng cũng như không trừu tượng.

• Nếu một lớp có dù chỉ một phương thức trừu tượng, lớp đó buộc phải là lớp trừu tượng.

• Một phương thức trừu tượng không có thân, khai báo phương thức đó kết thúc bằng dấu chấm phảy.

• Một lớp cụ thể phải cài đặt hoặc được thừa kế cài đặt của tất cả các phương thức trừu tượng.

19. Interface

- Interface là mức trừu tượng cao hơn lớp trừu tượng

- Chỉ bao gồm: phương thức trừu tượng, hằng số (static final)

- Mặc định là public

- Cú pháp: từ khóa interface và implements

- Java không cho phép đa kế thừa từ nhiều lớp cơ sở nhằm đảm bảo tính dễ hiểu và hạn chế xung đột nhưng có phép cài đặt đồng thời nhiều giao diện

20. Override (ghi đè) và overload (nạp chồng)

Override không được giảm quyền truy nhập so với phiên bản cha, danh sách tham số và giá trị trả về phải trùng khớp với phiên bản cha

Overloading cho phép ta tạo nhiều phiên bản của một phương thức, mỗi phiên bản chấp nhận một danh sách đối số khác nhau, kiểu trả về có thể khác nhau, có thể mở rộng hoặc hạn chế quyền truy nhập

21. Phân biệt kết tập và kế thừa

- Giống nhau: đều là kỹ thuật để tái sử dụng mã nguồn

- Kế thừa: tạo 1 lớp mới băng cách phát triển lớp đã có. Lớp con kết thừa dữ liệu và hành vi của lớp cha (VD: SinhVien kế thừa từ Person)

- Kết tập: tạp ra lớp mới là tập hợp các đối tượng của các lớp đã có. Lớp toàn thể có thể sử dụng dữ liệu và hành vi thông qua các đối tượng thành phần (VD: trong đối tượng LopHoc có đối tượng SinhVien)

22. Phân biệt interface và abstract class

Giống nhau:

- Không thể tạo đối tượng mới bằng toán tử new

Khác nhau:

|  |  |
| --- | --- |
| Interface | Abstract class |
| 1 class có thể implement nhiều interface | Không hỗ trợ đa kế thừa |
| Không thể định nghĩa code xử lý, chỉ có thể khai báo. Interface thì chứa method trống không có phần thân. Phần thân sẽ được code bởi những class implement interface đó | Có thể định nghĩa thân phương thức |
| Chỉ có phương thức abstract | Có các phương thức abstract và non-abstract |
| Trong khi Interface thì không thể implement class abstract | Class abstract có thể implement các Interface |
| Các thành phần mặc định là public | Các phương thức, thuộc tính có thể là public, private, default, protected |
| Class cài đặt Interface phải override hết tất cả các method có trong interface. | Class kế thừa chi phải override lại các abstract method |

23. Phân biệt lập trình hướng đối tượng và lập trình hướng cấu trúc

|  |  |
| --- | --- |
| Hướng đối tượng | Hướng cấu trúc |
| Chia chường trình thành các đối tượng | Chia chương trình thành các hàm |
| OOP chia ra các modifier: Private, Public, Protected, Default | Không |
| Các chức năng có thể chạy đồng thời | Các hàm, chức năng chạy lần lượt |
| Dữ liệu và hàm của một đối tượng giống như một thành phần riêng biệt và bị hạn chế truy cập bởi các đối tượng khác | Dữ liệu và hàm của một đối tượng giống như một thành phần riêng biệt và bị hạn chế truy cập bởi các đối tượng khác |
| OOP bảo mật hơn nhờ giới hạn truy cập dữ liệu | Kém hơn |
| OOP chú ý vào dữ liệu | Chú ý đến việc xây dựng các hàm và thuật toán |
| Hướng tiếp cận khi thiết kế chương trình: từ dưới lên | Hướng tiếp cận khi thiết kế chương trình: từ trên xuống |

24. Từ khóa super

Từ khóa super giúp tham chiếu đến lớp cha gần nhất của nó