ÔN TẬP CUỐI KỲ OOP

Câu 1: So sánh OOP với lập trình cấu trúc?

OOP	DSA
- Bottom-up: từ vấn đề nhỏ tổng kết lại thành bài toán lớn - Chú trọng dữ liệu - *Ưu điểm:* che giấu dữ liệu, tái sử dụng mã nguồn, dễ mở rộng - *Khuyết điểm:* cấu trúc phức tạp, khó theo dõi luồng dữ liệu, chỉ áp dụng cho hệ thống lớn	- Top-down: chia bài toàn lớn thành các bài toán con sau đó giải quyết các bài toán con - Chú trọng thuật toán - *Ưu điểm:* cấu trúc rõ ràng, đơn giản, dễ bảo trì và quản lý - *Khuyết điểm:* khó quản lý sự thay đổi dữ liệu, khó mở rộng, không tái sử dụng

Câu 2: Trình bày đặc điểm quan trọng của OOP?

- **Tính trừu tượng hóa:** tập trung vào nhựng gì cần thiết, là quá trình tổng quát hóa, quá trình ***quan trọng nhất***, có trừu tượng hóa về mặt dữ liệu (thuộc tính) và trừu tượng hóa về mặt hành vi (phương thức).
- **Tính đóng gói:** gom tất cả thuộc tính và hành vi vào 1 lớp ***giúp che giấu dữ liệu***.
- **Tính kế thừa:** gom thành phần giống nhau của các lớp để tạo lớp mới, giúp ***tái sử dụng*** mở rộng chương trình, bảo trì dễ dàng hơn.
- **Tính đa hình**: 1 đối tượng có nhiều hình thức, các đối tượng khác nhau thực hiện cùng 1 hành động cho ra kết quả khác nhau, thể hiện qua ***nạp chồng*** và ***ghi đè***.

Câu 3: Tại sao OOP là lập trình dữ liệu?

- Do phân tích bài toàn hướng đối tượng phải thông qua phân tích dữ liệu, sau đó trừu tượng hóa.

Câu 4: Nêu khái niệm Constructor, Destructor. Phân biệt Constructor mặc định và Contructor khác.

Constructor	- **Constructor** là hàm dùng để khởi tạo giá trị cho đối tượng. Được khai báo như một phương thức, tên phương thức trùng với tên lớp nhưng không có kiểu dữ liệu trả về. Một class có thể có nhiều constructor. Constructor phải có thuộc tính public. - **Constructor: không kế thừa**, được gọi từ **cha đến con**, được gọi đầu tiên trùng tên lớp.
Destructor	- **Destructor** là hàm dùng để hủy đối tượng khi hết phạm vi sử dụng. Được khai báo như một phương thức, tên phương thức trùng với tên lớp và có dấu ~ ở trước. Không có kiểu dữ liệu trả về. Destructor phải được khai báo ở nhãn public và chỉ có duy nhất 1 destructor trong class. - **Destructor**: đóng gói, giải phóng dữ liệu, **tránh rò rỉ** bộ nhớ (leak memory), **không kế thừa**, được gọi từ **con lên cha**.

- **Tại sao java không có destructor ?** ➔ Do có Garbage collector
- **Sự khác nhau giữa constructor và setter** ➔ Khác nhau ở thời điểm gọi, lần đầu tiên gọi thì dùng constructor.

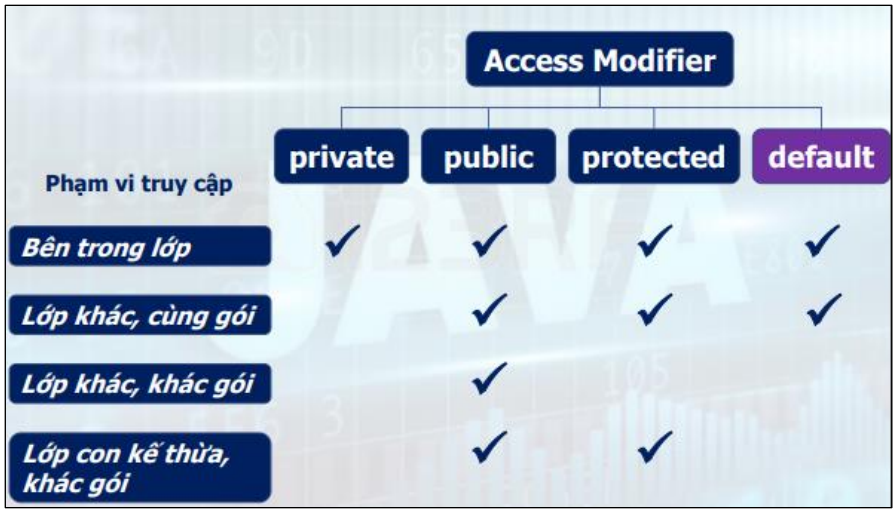
Phân biệt Constructor và Destructor	
Constructor	**Destructor**
- Không có kiểu dữ liệu trả về - Khai báo ở nhãn public - Được gọi tự động	
Tên trùng tên lớp	Tên trùng tên lớp nhưng thêm ~ ở trước
Khởi tạo giá trị cho đối tượng	Hủy đối tượng
Có nhiều constructor	Chỉ có 1 destructor

Phân biệt Constructor mặc định và constructor khác	
Constructor mặc định	**Constructor sao chép hoặc contructorcó tham số đầu vào**
- Được khai báo như một phương thức có tên trùng tên lớp - Là hàm dùng để khởi tạo giá trị cho đối tượng - Không có kiểu dữ liệu trả về	
Không có tham số truyền vào hoặc tất cả tham số điều có giá trị mặc nhiên.	Có một hoặc nhiều tham số đầu vào.
Khi khởi tạo đối tượng thì không cần truyền tham số vào	Khi khai báo đối tượng thì phải truyền tham số nếu không trình biên dịch sẽ dùng constructor mặc định.
Khởi tạo các đối tượng có cùng dữ liệu được khai báo trong constructor mặc định	Khởi tạo những đối tượng khác dữ liệu tùy thuộc vào tham số mà ta truyền vào
Chỉ có 1 constructor mặc định	Có thể có nhiều constructor có tham số

Câu 5: Class và Object

Class ⟶ **Object** | **Object** ⟶ **Class**

Câu 6: ACCSESS MODIFIER: **public > protected > default > private**



- **private:** bên trong lớp

- **public:** bên trong lớp, lớp khác cùng gói, lớp con kế thừa khác gói

- **protected:** bên trong lớp, lớp khác cùng gói, lớp con kế thừa khác gói

- **defaul:** bên trong lớp, lớp khác cùng gói

Câu 7: So sánh static với biến thường

Static	Non-static
- Dùng chung. - Được tạo liền đầu khi lớp được nạp. - Không được truy cập thành viên khác tĩnh.	- Dùng riêng. - Tạo mỗi lần khi có đối tượng mới. - Được truy cập, sử dụng thành viên tĩnh.
- Được truy cập thông qua tên lớp - Có thể được truy cập bởi phương thức tĩnh hoặc phương thức bình thường - Giảm dung lượng bộ nhớ được sử dụng - Được chia sẻ giữa tất cả instance trong 1 lớp - Như 1 biến toàn cục, dùng được cho tất cả phương thức	- Được truy cập thông qua instance của lớp - Không thể truy cập trong phương thức tĩnh - Không làm giảm dung lượng bộ nhớ - Cụ thể cho instance của 1 lớp - Như 1 biến cục bộ, chỉ đc truy cập thông qua instance của một lớp

- **Phương thức tĩnh (static):** Được kế thừa, không ghi đè

Câu 8: Phân biệt this và super?

This	Super
- Tham chiếu tới biến của lớp hiện tại - Gọi phương thức của lớp hiện tại - Gọi constructor của lớp hiện tại - Trả về instance của lớp hiện tại - Được truyền như 1 tham số trong phương thức, hay trong constructor	- Gọi trực tiếp constructor của lớp cha gần nhất - Gọi trực tiếp biến của lớp cha gần nhất - Gọi trực tiếp phương thức của lớp cha gần nhất

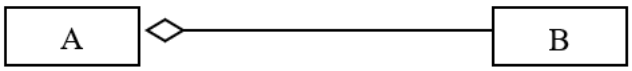
Câu 9: Nạp chồng và ghi đè

Ghi đè (overriding)	Nạp chồng (overloading)
- Dùng trong kế thừa	- Dùng trong lớp bất kì
- Phương thức cùng tên, cùng danh sách tham số, cùng kiểu dữ liệu trả về.	- Phương thức cùng tên, khác danh sách tham số
- Phạm vi truy cập không hạn chế hơn của cha	- Không có phạm vi truy cập

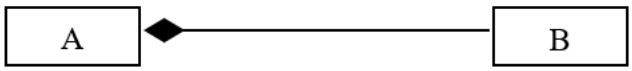
Câu 10: Quan hệ giữa 2 lớp

- **Quan hệ Association**

- **Quan hệ Aggregation**



- **Quan hệ Composition**



Câu 11: Kế thừa

- Java không hỗ trợ đa kế thừa.
- Không được phép kế thừa lớp final.
- Phương thức khởi tạo không được kế thừa ở lớp con.

Câu 12: Final

- Biến final: hằng số
- Phương thức final: không ghi đè phương thức final.
- Lớp final: không thể kế thừa lớp final.

Câu 13: **Lớp object**, các phương thức gồm có:

+ toString(): trả về chuỗi đại diện cho đối tượng.
+ equals(Object obj): kiểm tra hai đối tượng có bằng nhau.
+ hashCode(): trả về mã băm của đối tượng.
+ clone(): sao chép đối tượng ra đối tượng mới.
+ getClass(): trả về lớp mà đối tượng được tạo ra.

Câu 14: Trừu tượng (abstract) cần ghi đè

- Lớp trừu tượng không được tạo đối tượng
- Phương thức trừu tượng
- Lớp trừu tượng chứa phương thức trừu tượng

Câu 15: Kết hợp

abstract + static ➔ không được
abstract + final ➔ không được
abstract + defaul ➔ được
static + final ➔ được

Câu 16: Giao diện (interface)

- Không có phương thức khởi tạo
- Không được phép kế thừa lớp, các giao diện được phép kế thừa nhau
- Không thể tạo thể hiện từ giao diện

Câu 16: So sánh lớp Abstract & Interface

Abstract	**Interface**
- Không được tạo đối tượng - Đều chứa phương thức trừu tượng	
- Đơn kế thừa (1 lớp chỉ kế thừa 1 abstract class)	- Đa hiện thực (1 lớp hiện thực nhiều interface)
- Có constructor, static	- Không có constructor
- Có đầy đủ phương thức/thuộc tính của phương thức thường và có phương thức trừu tượng	- Chỉ có phương thức trừu tượng, nếu có thuộc tính thì chỉ là final và static

➢ **Tại sao có lớp abstract còn có interface?** Vì lớp trừu tượng chỉ có đơn kế thừa, mục tiêu của interface là đa hiện thực.

Câu 17: Các thành viên tĩnh sử dụng làm gì? Nó hoạt động thế nào trong lớp và trong quan hệ kế thừa?

- Thuộc tính tĩnh và phương thức tĩnh dùng chung cho tất cả các đối tượng lớp.
- Chúng được sử dụng thông qua tên lớp mà không cần tạo đối tượng.
- Trong phương thức tĩnh chỉ truy xuất được các thành viên tĩnh của lớp.
- Phương thức tĩnh được phép kế thừa ở lớp con nhưng không được ghi đè.

Cau 18: Trình bày cơ chế hoạt động của phương thức khởi tạo trong quan hệ kế thừa?

- Phương thức khởi tạo không được kế thừa ở lớp con.
- Thứ tự các phương thức khởi tạo được gọi là từ các lớp cha trước rồi mới đến lớp con.
- Các phương thức khởi tạo lớp con phải gọi phương thức khởi tạo lớp cha, nếu không Java sẽ ngầm định gọi phương thức khởi tạo không tham số của lớp cha.

Câu 19: Đa hình là gì? Nó được thể hiện thông qua cơ chế nào?

- Đa hình là khả năng của 1 đối tượng có thể thực hiện 1 tác vụ theo nhiều cách khác nhau.
- Đa hình được thể hiện thông qua cơ chế nạp chồng (overloading) và ghi đè (overriding).

PHÁT BIỂU ĐÚNG/SAI			
1	**Java là ngôn ngữ biên dịch và thông dịch**	**17**	**Không thể ghi đè phương thức khởi tạo.**
2	**Java là ngôn ngữ vừa hỗ trợ lập trình thủ tục, vừa hỗ trợ lập trình hướng đối tượng.**	**18**	**Khi lớp B kế thừa lớp A, nó còn ngầm định kế thừa lớp Object.**
3	**Chương trình Java có thể chạy trên bất kỳ nền tảng nào mà không cần biên dịch lại.**	**19**	**Phương thức private của lớp cha không được kế thừa ở lớp con.**
4	**Ngôn ngữ Java hỗ trợ khả năng đa kế thừa như trong C++**	**20**	**Phương thức private không thể được viết đè (overriding).**
5	**Phương thức khởi tạo không có kiểu dữ liệu trả về.**	**21**	**Phương thức static không thể được viết đè (overriding).**
6	**Phạm vi truy cập của phương thức khởi tạo phải là public.**	**22**	**Phương thức static của lớp cha không được kế thừa ở lớp con.**
7	**Tên ph.thức khởi tạo phải trùng với tên lớp.**	**23**	**Trong Java, 1 lớp có tối đa 1 lớp cha trực tiếp.**
8	**Lớp bắt buộc phải khai báo ít nhất một phương thức khởi tạo.**	**24**	**Lớp trừu tượng chỉ chứa các phương thức trừu tượng.**
9	**Một lớp có thể có nhiều phương thức khởi tạo.**	**25**	**Lớp trừu tượng không có phương thức khởi tạo.**
10	**Phạm vi truy cập mặc định của các thuộc tính và phương thức là private.**	**26**	**Các lớp trừu tượng không thể kế thừa lẫn nhau.**
11	**Phương thức thông thường không được phép truy cập các thành viên tĩnh của lớp.**	**27**	**Trong lớp trừu tượng không thể khai báo phương thức tĩnh.**

12	**Các phương thức nạp chồng phải cùng tên, khác kiểu dữ liệu trả về.**	**28**	**Lớp không có phương thức trừu tượng nào không thể khai báo là lớp trừu tượng.**
13	**Phương thức khởi tạo không thể khai báo là tĩnh.**	**29**	**Phương thức trừu tượng có thể khai báo là private hoặc final hoặc static.**
14	**Trong một lớp muốn sử dụng lớp khác phải import lớp đó trước khi sử dung.**	**30**	**Lớp trừu tượng bắt buộc phải có ít nhất một phương thức trừu tượng.**
15	**Một lớp có thể kế thừa nhiều lớp cùng lúc.**	**31**	**Các lớp trừu tượng có thể kế thừa nhau.**
16	**Phương thức tĩnh không được phép kế thừa ở lớp con.**	**32**	**Lớp con không thể truy cập vào phương thức public abstract của lớp cha.**
		33	**Lớp trừu tượng có thể kế thừa lớp thường**