Menu

[**Câu 1 : Sự khác nhau giữa override và overload?** 2](#_Toc81294828)

[**Câu 2: Lớp final là gì ?** 2](#_Toc81294829)

[**Câu 3 : Final Varibale (biến hằng)?** 2](#_Toc81294830)

[**Câu 4: final method?** 2](#_Toc81294831)

[**Câu 5: Đặc điểm biến static?** 2](#_Toc81294832)

[**Câu 6: Đặc điểm hàm static** 2](#_Toc81294833)

[**Câu 7: Sự khác biệt interface và lớp abstract ?** 2](#_Toc81294834)

[**Câu 8: Phân biệt public, private, protected** 3](#_Toc81294835)

[**Câu 9: Có bắt buộc sau khối try phải là khối catch hay không?** 3](#_Toc81294836)

[**Câu 10: Tính chất của OPP** 3](#_Toc81294837)

[**Câu 11: Có lớp static ko?** 3](#_Toc81294838)

[**Câu 12: Sử dụng từ khóa supper khi nào?** 4](#_Toc81294839)

[**Câu 13: Câu hỏi: Tại sao phương thức main trong Java là static?** 4](#_Toc81294840)

[**Câu 14: sự khác nhau giữa session và cokki** 4](#_Toc81294841)

[**Câu 15 : Triển khai 1 class đảm bảo tính đóng gói thì ntn?** 4](#_Toc81294842)

[**Câu 17: Phân biệt Stack và Queue** 4](#_Toc81294843)

[**Câu 18: List<Student> sang Set thì size thế nào?** 4](#_Toc81294844)

[**Câu 19:**  **Các thành phần trong collection framework java** 4](#_Toc81294845)

[**Câu 20:** **Tại sao cần phải bắt lỗi Expression?** 5](#_Toc81294846)

[**Câu 21:** **Eception với Error** 5](#_Toc81294847)

[**Câu 22. LinkedList trong java** 5](#_Toc81294848)

[**Câu 23. Xắp xếp trong Java dùng Comparator, Comparable?** 5](#_Toc81294849)

[**Câu 24. String trong Java** 5](#_Toc81294850)

[**Câu 25. Mối quan hệ is-a, has-a** 5](#_Toc81294851)

[**Câu 26. Cách so sánh 2 object bằng nhau:** 5](#_Toc81294852)

[**Câu 27. So sánh tham trị và tham chiếu.** 5](#_Toc81294853)

[**Câu 28. try - with resource ?** 5](#_Toc81294854)

[**Câu 29. Đơn kế thừa và đa kế thừa ?** 5](#_Toc81294855)

[**Câu 30.**  **Đối tượng là gì ?** 5](#_Toc81294856)

[**Câu 31. Phân biệt statement và preparedstatement trong java** 5](#_Toc81294857)

[**Câu 32. Stateless và Stateful** 6](#_Toc81294858)

[**Câu 33. Exception, trycatch** 6](#_Toc81294859)

# **Câu 1 : Sự khác nhau giữa override và overload?**

* Override là ghi đè phương thức, các phương thức này được viết ở 1 lớp khác (cụ thể là lớp cha). Nhưng do không phải lớp con nào cx phù hợp với lớp cha nên lớp con cần phải overtide lại để cho phù hợp.
* Overload là nạp chồng phương thức, các phương thức có tham số truyền vào khác nhau và có thể viết ở cùng 1 lớp

# **Câu 2: Lớp final là gì ?**

* Là lớp mà trong đó các phương thức được tạo ra không để ghi đè, và final ko thể bị kế thừa (tức là không lớp nào kế thừa được nó nữa.

# **Câu 3 : Final Varibale (biến hằng)?**

* Biến hằng là biến không thể thay đổi giá trị (nên viết tên biến bằng chữ in hoa)

# **Câu 4: final method?**

* Là phương thức mà lớp con không thể ghi đè.
* Nếu class ko phải final thì có thể thừa kế, nhưng ko thể ghi đề phương thức final của class đó

# **Câu 5: Đặc điểm biến static?**

* Gọi là biến TĨNH
* Chỉ phụ thuộc vào lớp mà không phụ thuộc vào đối tượng.
* Biến static chỉ khởi tạo 1 lần khi chương trình bắt đầu thực thi.
* Biến static có thể truy cập trực tiếp bằng tên class mà không cần bất kỳ đối tượng nào.
* Cú pháp : tên\_class.tên\_biến\_static

# **Câu 6: Đặc điểm hàm static**

* Chỉ phụ thuộc vào lớp mà không phụ thuộc vào đối tượng.
* Phương thức chỉ có thể truy cập vào các thành phần static mà không thể truy cập vào các thành phần không static

# **Câu 7: Sự khác biệt interface và lớp abstract ?**

|  |  |
| --- | --- |
| InterFace  - Hỗ trợ đa thừa kế  - Chỉ có phương thức trừu tượng  - Chỉ có biến static, final | Abstact  - Không hỗ trợ đa thừa kế  - Có thể có phương thức trừ tượng và không trừu tương  Có biến static non static, final và non final |

**Câu 8: Phân biệt public, private, protected**

* Các thuộc tính hoặc method có thể truy xuất ra bên ngoài
* Các thuộc tính các phương thức không thể truy xuất ra bên ngoài class, chỉ được gọi trong phạm vi class
* Các thuộc tính, method không thể truy xuất ra bên ngoài class, nó chỉ được gọi trong class và class kế thừa

# **Câu 9: Có bắt buộc sau khối try phải là khối catch hay không?**

* Không bắt buộc
* Nhưng bắt buộc phải có 1 trong 2 khối catch hoặc finally ( được sử dụng để thực thi các lệnh quan trọng như đóng kết nối, đóng cá stream,... Dù câu lệnh trong try hay cath thì finally luôn luôn chạy.)

# **Câu 10: Tính chất của OPP**

* Lập trình hướng đối tượng là một kĩ thuật lập trình cho phép lập trình viên tạo ra các đối tượng trong code và trừu tượng hóa chúng.

Có 4 tính chất của lập trình hướng đối tượng

- Tính đóng gói : Nhằm bảo vệ đối tượng không bị truy cập từ code bên ngoài vào để thay đổi các giá trị của thuộc tính hay truy cập trực tiếp. Muốn cho phép truy cập các giá trị của đối tượng, thì phải tùy theo sự đồng ý của người viết ra lớp của đối tượng đó. Đảm bảo sự bảo mật, toàn vẹn dữ liệu của đối tượng.

\* Ví dụ ở class khác muốn set giá trị cho private String name, thì phải viết 1 method public setter, chứ ko thay đổi trực tiếp được

* Tính kế thừa : là kỹ thuật cho phép kế thừa lại những tính năng mà 1 đối tượng khác đã có, giúp tránh việc code lặp lại dư thừa mà chỉ để xử lý công việc tương tự
* Tính trừu tượng: là chỉ nói ra tính năng của người dùng, các khái niệm sẽ được định nghĩa trong quá trình phát triển, cho phép người lập trung tập trung cốt lõi cần thiết, thay vì quan tâm sự phức tạp bên trong.
* Tính đa hình: là khả năng 1 đối tượng có thể thực hiện một tác vụ theo nhiều cách khác nhau. Được thể hiện rõ nhất qua việc gọi phương thức, các phương thức hoàn toàn có thể giống nhau, nhưng xử lý logic thì khác nhau.

**Câu 11: Có lớp static ko?**

* Có
* Một class được đặt là static chỉ khi nó là một nested class (tức nằm trong 1 lớp khác)

# **Câu 12: Sử dụng từ khóa supper khi nào?**

* Supper được sử dụng để tham chiếu trực tiếp đến đối tượng gần nhất của lớp cha.
* Nếu method hay thuộc tính ghi đè lớp cha , nếu muốn gọi method hay biến của lớp cha, ta phải dùng super.method
* Super() : gọi trực tiếp constructor của class cha.

# **Câu 13: Câu hỏi: Tại sao phương thức main trong Java là static?**

* Bởi vì không cần thiết phải tạo đối tượng để gọi phương thức static. Nếu nó là phương thức non-static, JVM đầu tiên tạo đối tượng và sau đó gọi phương thức main() mà có thể gây ra vấn đề về cấp phát bộ nhớ bộ nhớ phụ.

# **Câu 14: sự khác nhau giữa session và cokki**

* Là Session lư trên server còn cokki thì lưu trên máy người dùng.

# **Câu 15 : Triển khai 1 class đảm bảo tính đóng gói thì ntn?**

Cần phải khai báo private cho các thuộc tính.

**Câu 16: Trả lời về haschCode , equals**

+ equals được thiết kế để so sánh hai đối tượng về mặt ngữ nghĩa ( bằng cách so sánh các dữ liệu thành viên cảu lớp)

+ hashcode : khi so sánh thì nó sẽ trả về mộ giá trị.

# **Câu 17: Phân biệt Stack và Queue**

+ Stack là vào sau ra trước

+ Queue là vào trước ra trước

# **Câu 18: List<Student> sang Set thì size thế nào?**

# **Câu 19:** **Các thành phần trong collection framework java**

+ collection interface

+ set interface

+ list interface

+ queue interface

+ map interface

# **Câu 20:** **Tại sao cần phải bắt lỗi Expression?**

# **Câu 21:** **Eception với Error**

+ Eception là những lỗi phát sinh mà chúng ta có thể lường trước và bắt lỗi đc

+ Error là những lỗi phát sinh trong quá trình phát triển dự án mà chúng ta không thể lường trước đc.

# **Câu 22. LinkedList trong java**

ok, cần nói rõ hơn các interface kế thừa List, Queue, và mô tả rõ liên kết kép

# **Câu 23. Xắp xếp trong Java dùng Comparator, Comparable?**

+ comparator khi so sánh thì không cần implement Comparator. Phương thức compare() được sử dụng để so sánh giữa hai đối tượng obj1 và obj2

+ Comparable(compareto) khi so sánh cần implement interface Comparable và đc cung cấp 1 phương thức để so sánh.

# **Câu 24. String trong Java**

khởi tạo và hiểu String literal( khởi tạo bởi dấu “ ”) và String new

* String literal khi khởi tạo thì nó sẽ kiểm tra String pool đã tồn tại hay chưa nếu tồn tại rồi thì nó sẽ tham chiếu đến instate đã tạo trong pool và nếu chưa tồn tại thì sẽ tạo ra 1 instate mới ở trong pool.
* String new : cú pháp String s = new String().

# **Câu 25. Mối quan hệ is-a, has-a**

# **Câu 26. Cách so sánh 2 object bằng nhau:**

Muốn đối tượng bằng nhau khi và chỉ khi: equals trả về true và hashcode trả về cùng một giá trị.

# **Câu 27. So sánh tham trị và tham chiếu.**

+ Tham trị (Pass by value) : Là khi khởi tạo mk sẽ tạo ra một giá trị mới bằng cách coppy giá trị gốc và chỉ thao tác với giá trị coppy và giữ nguyên giá trị gốc.

+ Tham chiếu (Pass by reference**) :** Khi bạn thay đổi giá trị trong hàm thì ngoài hàm cx bị thay đổi**.** Nó giống như việc bạn chuyển đúng địa chỉ đó vào hàm.

# **Câu 28. try - with resource ?**

Câu lệnh try-with-resources là câu lệnh khai báo một hoặc nhiều tài nguyên. Một tài nguyên là một đối tượng mà phải đóng sau khi chương trình kết thúc với nó

# **Câu 29. Đơn kế thừa và đa kế thừa ?**

* Đơn kế thừa là 1 class chỉ đc kế thừa đc 1 class khác (1 con thì chỉ đc extend 1 cha)
* Đa kế thừa: 1 class có thể kế thừa từ nhiều class khác ( nhưng bắt buộc phải sử dụng interface )

# **Câu 30.** **Đối tượng là gì ?**

# **Câu 31. Phân biệt statement và preparedstatement trong java**

+ Statement : dùng để thực thi các câu lệnh SQL tĩnh (không có tham số truyền vào trong thời gian runtime)

+ Preparedstatement : Đùng để thực thi truy vấn các câu SQL động hoặc có tham số. Nó thừa kế từ statement nhưng nó cho phép truyền các tham số vào câu lệnh SQL trong thời gian runtime

# **Câu 32. Stateless và Stateful**

+ Stateless là một dạng web nhưng không lưu dữ liệu của người dùng. Khi client gửi dữ liệu lên server, server thực thi xong sẽ trả về kết quả thì mối quan hệ giữa client và server sẽ bị cắt đứt.

+ Stateful là 1 web cần server lưu trữ lại dữ liệu của client. Data đc lưu ở phía server và khi client thao tác lại thì phía server sẽ trả về dữ liệu từ lần trước thao tác. Có 4 cách lưu data của client khi xây dựng Web Application bao gồm : URL Rewriter , form , cokki ,HTTP Session.

# **Câu 33. Exception, trycatch**

+ Exception : khi chạy hàm main nếu gặp lỗi thì trương trình sẽ dừng lại và báo lỗi.

+ try catch: Là bắt được lỗi và ném ra lỗi nhưng trương trình vẫn sẽ được thực thi.

+ queue interface

+ map interface

II.SQL

**1.Nắm được kiến thức về thực thể**

* 1. **Thực thể trong SQL.**

**a - Thực thể và tập thực thể**

* Thực thể là một đối tượng trong thế giới thực. Một nhóm bao gồm các thực thể tương tự nhau tạo thành một tập thực thể
* Ví dụ: “Quản lý các dự án của công ty”
* Một nhân viên là một thực thể
* Tập hợp các nhân viên là tập thực thể

**b - Thuộc tính:**

Mỗi tập thực thể có một tập các tính chất đặc trưng, mỗi tính chất đặc trưng này gọi là thuộc tính của tập thực thể. Ứng với mỗi thuộc tính có một tập các giá trị cho thuộc tính đó gọi là miền giá trị.

**Các loại giá trị của thuộc tính:**

* Đơn trị: các thuộc tính có giá trị duy nhất cho một thực thể (VD: số CMND, …)
* Đa trị: các thuộc tính có một tập giá trị cho cùng một thực thể (VD: bằng cấp, …)
* Suy diễn được (năm sinh <----> tuổi)

Class là khuôn mẫu của đối tượng có các thuộc tính chung,

Ví dụ class smartphone, còn object đc tạo ra từ class như iphone, samsung

ok

- Nắm được kiến thức về Join

**2.Join trong sql**

- Join để lấy dữ liệu từ nhiều bảng xảy ra khi 2 hoặc nhiều bảng đc kết nối với nhau trong một lệnh SQL.

- Các loại join thường dùng.

+ **INNER JOIN : còn gọi là hình thức kết nối đơn giản.**

**Cú pháp : Select cot from bang1 INNER JOIN bang2 on bang1.cot = bang2.cot;**

**Kết quả trả về : là những bản ghi chung giữa bang1 và bang2.**

**+ Left outer JOIN hay left join.**

**Cú pháp: SELECT COT FROM BANG1 LEFT JOIN BANG2 ON BANG1.COT = BANG2.COT**

**Kết quả trả về : Trả về các hàm nằm ở bảng bên trái trong điều kiện ON và chỉ lấy ở các bảng còn lại những hàm đáp ứng được điều kiện đặt ra .Trả về gòm bảng 1 và các bản ghi nằm trong bảng 2 giao với bảng 1.**

**+ RIGHT JOIN.**

**Tương tự như bản LEFT JOIN**

**+ Full join.**

**Trả về tất cả bản ghi**

**3.Nắm được Union và Union All**

* UNION đc dùng để kết hợp 2 bộ kết quả từ 2 hoặc nhiều SELECT. Nó sẽ xóa các hàng trùng trong các lệnh SELECT này. Mỗi lệnh SELECT trong toán tử UNION phải cùng số cột trong bộ kết quả với dữ liệu tương ứng.
* UNION ALL trả về tất cả các hàng từ truy vấn và không xóa các hàng trùng lặp.

4.Đánh Index là gì?

Index là một chỉ mục nó giúp cho câu lệnh SELECT trở lên nhanh hơn và đỡ tốn thời gian tìm kiếm hơn.

- Trigger và Sp

+ Trigger sẽ được tự động thực thi khi xảy ra một sự kiện trên một bảng (Các sự kiện bao gồm insert, update, delete)

+

- Toán tử i++ và ++i

5. Khác nhau đó là thứ tự hoạt động tăng lên

A = i++ : gán i cho A trước (gán giá trị khởi tạo trước) rồi mới cộng i lên 1

A= ++i : tăng i lên trước rồi mới gán cho

6. Thứ tự của các mệnh đề trong câu lệnh SQL Select.

- Select, form , where , Groub by , Having, orderby