|  |  |
| --- | --- |
| What is JDK, JRE and JVM ? | * **JDK (java developtment kit)** : là các gói phần mềm tải xuống để tạo ra các ứng dụng * **JRE (Java Runtime Evironment) :** chứa các thư lớp java, trình tải java và máy ảo. . * JVM là 1 hệ thống phần mềm đang thực chương trình java trực thì **JRE** **là hệ thống trên đĩa lấy mã java, kết hợp nó với thư cần thiết để khởi động JVM** để thực thi * JVM ( Java virtual machine) : máy ảo, sẽ cho chương trình java chạy trên mọi thiết hoặc hệ điều , và để qly tối ưu hóa bộ nhớ * có vai trò thực thi mã java thành bytecode * <https://viblo.asia/p/jvm-jdk-jre-co-gi-khac-biet-giua-chung-oOVlYBXV58W> |
| What is the difference between Object (Instance) and Class ? | * Object là những thể hiện của class * khởi tạo object với từ khóa “new” * Class định nghĩa cho tất cả các phương thức, thuộc cần của 1 object * Object phải phụ thuộc , tất cả object thuộc class đều có cùng thuộc tính và phương thức |
| What is the difference between == vs equals() ? | * **“==”** (toán tử) : so sánh tham chiếu, cả 2 biến đề cập đến 1 đối tượng * “**equals**()” (method) : so sánh các giá trị trong đối * http://hungpv7.blogspot.com/2017/07/phan-biet-toan-tu-va-phuong-thuc-equals.html |
| What is the instanceof operator ? | * **Instanceof** : là toán tử để kiểm tra xem đối có phải là thể của một kiểu dữ liệu cụ thể ko * https://www.geeksforgeeks.org/instanceof-operator-vs-isinstance-method-in-java/ |
| What is the difference between \* and % operators ? | * “\*” là toans tử nhân * modulo là phép toán tìm số dư * ví dụ : 5:2 = 2 dư 1(modulo) |
| What is the difference between expr++ and ++expr | * cả 2 đều sẽ tăng lên 1 nếu như thực thi độc lập * ++i : sẽ tăng giá trị của i trả về giá trị * i++ : tăng giá trị i, nhưng trả về kqua trước khi tăng i |
| What is the Object class ? | * Mention above |
|  |  |
| What is difference between Overriding and Overloading ? | * Overloading : cho phép các method có cùng nhưng khác thuộc tính ( khác theo số lượng tham đầu vào hoặc loại tham số đầu vào). Method ten giong nhau, nhung doi’ so truyen vao khac nhau, cung nhu la so luong truyen vao * Override : Ghi đè cho phép một lớp con hoặc lớp con cung bởi một lớp cao hơn hoặc lớp cha của nó * <https://www.geeksforgeeks.org/difference-between-method-overloading-and-method-overriding-in-java/> |
| What is static keyword ? the static can be variable, method, block and nested class ? | * Static keyword : khi một phương thúc,biến được khai tĩnh, nó có thể dc truy cập trước khi bất kỳ obj của lớp tạo * ham hoac bien nao declared static se la 1 lop * Blocks: nếu cần tính toán để khởi tạo các biến tĩnh , cta có thể khai báo khối dc thực thi chính xác 1 * Variables: khi một biến dc khai báo , thì một bản sao của biens dc gán sẽ chia sẻ giữa tất cả obj ở cấp lớp * Methods: khi phương thức kbao tĩnh thì nó cũng truy cập trước khi đối tượng hay class của nó dc tạo. nhưng có 1 số bất lợi   + method static chỉ có thể gọi trực tiếp các static khác   + chỉ có thể trực tiếp truy cập dữ liệu   + không thể refer tới this hay super   Nested clas laf 1 class khai bao trong class khac  Khong the access non-static data khac, no co the access static data |
| Why the main() method need to be static ? | * Vì từ khóa cho phép gọi chinhs mà ko cần tạo obj của trong đó phương thức chính dc định nghĩa |
| What is Final keyword ? | * khi 1 biến dc khai báo với từ khóa , giá trị của nó sẽ ko thể thay doi du o ba cu dau |
| Difference between Inheritance & Aggregation ?  **public class** testing {  **void** play()  {  System.***out***.println(**"play"**);  }  **void** eat()  {  System.***out***.println(**"eat"**);  }  **void** walk()  {  System.***out***.println(**"walk"**);  }  **public static void** main(String[] args) {  testing t = **new** testing();  cat c = **new** cat();  dog d = **new** dog();  }  }  **class** dog **extends** testing  {  **void** bark()  {  System.***out***.println(**"bark"**);  }  }  **class** cat **extends** testing {  **void** purr()  {  System.***out***.println(**"purr"**);  }  } | * Ke thua * tap hop : la quan he giua toan the va bo phan, trong do mot lop bieu dien cai lon hon, con lop kia bieu dien cai nho hon |
| What is String Literal ? | * là obj dc hỗ trợ bên bởi một mảng char * chuỗi * String được lưu trữ trong ngăn , đòi hỏi space , rẻ hơn khi thao tác * la tap hop ky tu cua mot hcuoi |
| What is the difference between String, StringBuffer and StringBuilder ? | * String thì có độ dài cố định * Stringbuffer đại diện cho ký tự có thể phát triển và có thể ghi.(xử lý văn bản nhiều luồng) * StringBuilder: đại diện cho 1 chuỗi có thể thay đổi(xử lý văn bản 1 luongf) * <https://o7planning.org/vi/10217/huong-dan-java-string-stringbuffer-va-stringbuilder> * La 2 class trong java build san * toc do nhanh hon vi chi khai bao 1 lan String, sau do append lai |
| What is immutable class ? How to create an immutable class ? | * Lớp Bất biến : khi mà objdc tạo thì chugns ta ko thể thay đổi content * <https://www.geeksforgeeks.org/create-immutable-class-java/> * keyword final |
| Why String class is immutable ? | * vì các obj chuỗi lưu trong bộ nhớ chuỗi (String pool), vì các dữ liệu lưu trong bộ nhớ , luôn có rủi, hành động của 1 khách hàng sẽ ảnh hưởng tới toàn bộ. * **Ví dụ** : nếu 1 khách hàng thay đổi value “test” -> “TEST”, tất cả khách hàng khác cũng sẽ thấy thay đổi thế nên phải set chuỗi là bất biến |
| What is String intern() ? | * l**ưu trữ bản sao của mỗi giá chuỗi riêng , phải là bất biến.** * khi intern() dc thực thì nó sẽ ktra xem chuỗi có bằng với obj chuỗi này có trong String pool hay ko * <https://www.geeksforgeeks.org/interning-of-string/> |
| What is the difference between Set, List and Map ? | * là 3 đều là ìnterface * Khác nhau giữa list và set là **list** cho phép duplicate (Trùng lặp) còn **set** thì ko * Khi một map 2 đối tượng thực thể là key và value thì ko dc trùng * Set la java collection unique trong tap hop do * de lap set dung iteritor hoac for * List thì duy trì thứ tự chèn, còn set thì ko * List cho phép null và nhiều giá trị null vì có tính duplicate * Set thì chỉ cho phép duy nhất 1 null * <https://www.java67.com/2013/01/difference-between-set-list-and-map-in-java.html> |
| What is the difference between Set and HashSet, List and ArrayList, Map and HashMap ? | * Lớp Hashset thực hiện giao diện set, được hỗ trợ bởi hàm băm * cấu trúc dữ leieuj cơ bản cho hashset là hashtable * các giá trị trùng lặp ko cho phép * obj dc chèn dựa trên mã băm của chúng * null dc cho phép * <https://www.geeksforgeeks.org/hashset-in-java/> * Lisst and Array List * List la 1 interface, arraylist la 1 class collection tieu chuan * List su dung de tao list cua phan tu ma dc lien ket voi chi muc cua chung * Arraylist su dung de tao ra mang chứa các dối tượng * <https://www.geeksforgeeks.org/array-vs-arraylist-in-java/> * **Map and HashMap** * Map là một tập dữ liệu lưu dưới dạng key value, một map ko thể chứa những key trùng nhau, nhưng mỗi key có thể có ánh xạ tới nheieuf hơn 1 giá trị * map interface là method thực hiện các thao tác (get,put,remove,constantkey, size,empty) * Hash Map là 1 implement của map interface * hash map ko chứa duplicate key, cho phép null value và null key * là unorder collection   **Itorator trong lucs lap co the the xoa phan tu trong 1 list khi so sanh equals con` for thi ko** |
| What is the difference between Error and Exception ? | * Error : chir ra nhuwng vấn đề nghiêm trọng mà ứng dụng hợp lí ko nên nắm * Error là lỗi điều khiển ko thể phục hồi. * thuộc vè loại ko ktra và hay gặp trong lúc runtime * Exception : la cac dieu kien ma cac ung dung hop li co the nam bat va khi bi dung chuong trinh co the phuc hoi dc. * https://www.geeksforgeeks.org/errors-v-s-exceptions-in-java/ |
| What is the difference between Checked and Unchecked exception ? | * **Checked** : là các ngoại lệ xayr ra tại thời điểm biên, nếu trong code đưa ra 1 ngoại lệ dc ktra thì phương thức đó phải xử lý ngoại bằng cách sử dụng từ khóa throws * **Uncheck** laf cacs ngoaij le ko dc kiem tra o thoi dien bien dich. * Trong java thi Error va RuntimeException la class unchecked exception,moi thu khac co the “throws” deu co the checked * https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/ |
| How do we know if a class is checked or unchecked exception ? | * Checked can phai dc khai bao trong lenh “Throws” cua phuong thuc hoac ham nếu chúng có thẻ “throws” hoặc thực thi bởi phwuong thức * được sử dụng khi bạn mong muốn có 1 ngoại trong chương trình của và bạn muốn nguwoif ogij xử lý * Uncheck dc sử dụng khi bạn ko mong đợi nó xảy ra * https://stackoverflow.com/questions/46922986/how-to-know-if-my-code-is-using-a-checked-or-an-unchecked-exception |
| How to handle exception ? how try-catch-finally work ? | * **Exeption handling** là một trong những cơ chế mạnh để xử lý thời gian để có thể duy trì luồng thông thường của ứng dụng * có 5 loại handling : * try-catch-finally-throw-throws * <https://www.javatpoint.com/exception-handling-in-java> * try : chỉ định 1 nơi mà đặt mã ngoại, try phải đi kèm với catch hoặc finally * catch : để xử lý exception, phải đc bắt đầu bằng khối * finally : để thực thi mã qtrong của chương trình, nó dc thực thi cho dù một ngoại lệ có dc xử lý or ko * throw : sử dụng để loại 1 exception * throws : dùng để khai báo ngoại lệ, nó chỉ định rằng có thể xảy ra ngoại lệ trong method,luôn dc sử dụng với signature method |
| Can we create a custom exception ? When we should to create a custom exception ? | * Yes * <https://www.geeksforgeeks.org/g-fact-32-user-defined-custom-exception-in-java/> * chỉ nên custom exception nếu chắc chắn nó cung cấp một lợi ích so với các ngoại lệ tiêu chuẩn của java * <https://www.geeksforgeeks.org/g-fact-32-user-defined-custom-exception-in-java/> |
| Is there any constraint while declaring multiple catch statements ? | * dc nếu catch thứ 2 trở đi ko phải IOException , throws IOException, thì block catch thứ 2 sẽ ko bao giờ thực thi * <https://stackoverflow.com/questions/12583487/execution-of-multiple-catch-blocks> |
| ? Do we need to declare throws a RuntimeException ? | * NO * một nguwoif gọi method có throws runtime exception ko bị bắt buộc phải bắt ngoại lệ này * RE extend from RuntimeException hoặc ERROR * <https://www.javaworld.com/article/2077330/exceptions-don-t-get-thrown-for-a-loss.html> |
| What are the rules if we talk about methodoverriding with exception handling ? | * overriding method có thể đưua ra bất kỳ unchecked method nào, bất kể phương thức ghi đè có throws exception or not * nếu lớp extend từ base class throws 1 unchecked Ex → succees * base class ko có, lớp extend có checked EX → fail * base class and class extend có checked EX → succees * extend class throws 1 Exception Rộng hơn so sánh với base class → fail+ * <https://beginnersbook.com/2014/01/exception-handling-in-method-overriding-with-example/> |