|  |
| --- |
| JAVA là gì ?  Là ngôn ngữ lập trình hướng đối tượng OOP, thực thi được trên nhiều nền tảng và thiết bị.  Write once, run anywhere |
| JDK, JRE, JVM   * JDK là tool – bộ công cụ hổ trợ lập trình ứng dụng Java (Java Development Kit) * JRE là môi trường để thực thi ứng dụng Java(Java Runtime Environment) * JVM là máy ảo Java, thực thi mã java byte code(.class) ra ngôn ngữ máy   (Java Virtual Machine) |
| Thông dịch, biên dịch khác nhau như thế nào ?   * Biên dịch là dịch toàn bộ file 1 lần . Sau đó sử dụng kết quả biên dịch mà không cần biên dịch lần nữa.(\*.java 🡪\*.class). * Thông dịch là dịch từng dòng lệnh, muốn chạy lại thì phải chạy lại lần nữa.(JVM thông dịch \*.class ra ngôn ngữ máy) |
| JAVA là ngôn ngữ thông dịch hay biên dịch ?  Java là ngôn ngữ vừa thông dịch vừ biên dịch. JDE đóng vai trò biên dịch còn JVM đóng vai trò thông dịch. Vì vậy Java có thể chạy trên mọi nền tảng mà có JVM. |
| Các kiểu dữ liệu trong java, và giá trị mặc định khi khai báo.   * Kiểu nguyên thủy (8 loại):   + Byte (1byte) kiểu số nguyên. Giá trị nằm trong khoảng (-128 🡪 127)  +Short (2byte) kiểu số nguyên. Giá trị nằm trong khoảng (-32768 🡪 32767)  +int (4byte) kiểu số nguyên. Giá trị nằm trong khoảng ( 🡪)  +long (8byte) kiểu số nguyên. Giá trị nằm trong khoảng ( 🡪)  +float (4byte) kiểu số thực  +double (8byte) kiểu số thực  +boolean: bao gồm 2 giá trị là true và false  +char: kiểu kí tự Unicode có kích thước 2 byte. Có giá trị nhỏ nhất \u0000 ( < == > 0) và giá trị lớn nhất là \uffff (< == > 65535) |
| Có bao nhiêu lại mệnh đề if (4 mệnh đề)  If , if – else , if lồng nhau , if bậc thang |
| So sánh if và switch-case   * If có thể sử dụng để so sánh lớn hơn, nhỏ hơn….. Switch case chỉ có thể sử dụng so sánh bằng và khác nhau. * If thì mỗi câu lẹnh if có 1 biểu thức điều kiện , trả về giá trị True hoặc false. Switch case thì tất cả các trường hợp đều so sánh với giá trị với một biểu thức điều kiện duy nhất * If thì biểu thức điều kiện trả về giá trị kiểu boolean. Switch case thì biểu thức điều kiện trả về giá trị kiểu byte, short, char, int, hoặc string. * If thì chỉ có khối lệnh thực thi nếu điều kiện đúng. Switch case nếu điều kiện đúng mà không có câu lệnh break thì tất cả các khối lệnh ở phía sau cũng được thực thi. |
| Khi nào dùng for, while, do-while ?   * For : biết trước số vòng lặp * While : không biết trước số vòng lặp * Do-while : chạy trước 1 lần rồi mới kiểm tra |
| Phân biệt for-i và for-each   * For – I là in ra index và có thể duyệt bất cứ đâu, có thể duyệt từ đầu mảng đến cuối mảng hoặc ngược lại * For – each là in ra đúng giá trị và chỉ duyệt được từ đầu mảng đến cuối mảng. |
| Khác nhau giữa while, do-while. Cho ví dụ khi nào dùng ?   * while : điều kiện lặp sẽ được xét trước sau đó khối lệnh trong while mới được thực thi * do-while: khối lệnh sẽ được thực thi sau đó mới xét đến điều kiện lặp trong while |
| Break, Continue có tác dụng gì trong mệnh đề lặp ?   * Break: dùng để thoát khỏi vòng lặp . Nếu nhiều vòng lặp lồng vào nhau thì break sec thoát ra khỏi vòng lặp gần đó nhất * Continue: nhảy ra 1 phần của khối lệnh thực thi và bắt đầu vòng lặp mới, khi chương trình gặp lệnh continue thì tất cả các câu lệnh bên dưới đều bị bỏ qua ở vòng lặp hiện tại. |
| Trình bày các cách khởi tạo một mảng trong JAVA ?   * Có 2 cách : * elementType [] arrayName * elementType arrayName [] |
| Phần tử của mảng có thể dùng kiểu dữ liệu nào, và có giá trị mặc định là gì ?   * Nguyên thủy và đối tương * null |
| OOP là gì ?   * Là kỹ thuật lập trình nhiệm vụ ánh xạ các đối tượng thực tế sang các đối tượng trong lập trình. * Class là khuôn mẫu gom tất cả cá dữ liệu phương thức thuộc tính của 1 nhóm đối tượng liên quan đến nhau thành 1 lớp * Object là thể hiện của 1 lớp |
| Phân biệt class và object:   * Class là 1 kiểu dữ liệu được các lập trình viên tạo ra để mô hình hóa các đối tượng class sẽ mô tả các thuộc tính và hành vi chung của 1 nhóm các đối tượng. * Thành phần: phương thức , thuộc tính và contructor. * Object là đối tượng cụ thể thể hiện ta từ class, có các giá trị cụ thể của thuộc tính và hành vi. |
| Constructor là gì  Là phương thức để khởi tạo đối tượng |
| Cách khai báo constructor và đặc điểm constructor trong JAVA   * (khả năng truy cập) (tên phương thức) () {   Code  }   * Đặc điểm: * Tên của contructor trùng với tên class * Không có kiểu trả về * Trong 1 class có thể có nhiều contructor * Mặc định nếu không khai báo contructor thì java sẽ cung cấp 1 contructor mặc định không tham số(body rỗng) * Nếu có khai báo contructor có tham số thì contructor mặc định của java sẽ bị mất 🡪 phải định nghĩa lại * Có thể gọi contructor khác bằng từ khóa this. |
| Phân biệt constructor và method   |  |  | | --- | --- | | Contructor | Method | | * Khởi tạo trạng thái của đối tượng * Trùng với tên class * Không có kiểu trả về * Chỉ được gọi 1 lần khi khởi tạo đối tượng * Nếu không khai báo thì java cung cấp contructor mặc định | * Thể hiện hành động của 1 đối tượng * Có thể trùng hoặc không trùng với tên class * Có kiểu trả về * Có thể khởi tạo đối tượng nhiều lần * Java không cung cấp contructor mặc định | |
| Tính bao đóng là gì ?   * Là kỹ thuật ẩn dấu thông tin đối tượng, chỉ hiển thị những thông tin cần thiết. Được thể hiện rõ nhất trong 1 lớp. * Mục đích : Bảo vệ trạng thái bên trong của đối tượng – không cho phép truy cập trực tiếp thay đổi giá trị thuộc tính của đối tượng.   Giảm độ phức tạp khi phát triển phần mềm  Làm sao để thu được tính bao đóng trong java ?   * Cần thông qua: Access Modifier , Setter và getter |
| Tham trị, tham chiếu   * Tham trị(giá trị) là truyền vào kiểu nguyên thủy, giá trị của biến trước và sau khi truyền vào method là không thay đổi. * Tham chiếu (vị trí) là truyền vào kiểu đối tượng, giá trị của biến trước và sau khi truyền vào method có thể thay đổi |
| Từ khóa static dùng để làm gì ?   * Static dùng cho class, phương thức, thuộc tính * Mục đích: Xây dựng class tiện ích, sử dụng ngay mà không cần tạo đối tượng   Tạo thuộc tính và phương thức chung cho tât cả các đối tượng của class.   * Truy cập thành phần static trực tiếp thông qua class hoặc đối tượng |
| Ràng buộc khi sử dụng static   * Phương thức static chỉ gọi được phương thức static khác * Phương thức static chỉ truy xuất biến static. * Khởi tạo giá trị của biến static thông qua khối khởi tạo static |
| Các loại biến trong JAVA |
| Trình bày các loại access modifier, và phạm vi truy cập.   * Access Modifier là phạm vi truy cập * Các loại: |
| Kế thừa trong JAVA là gì ?   * Là kỹ thuật tạo ra lớp mới dựa trên cơ chế của lớp cũ. Mục đích tái sử dụng mã nguồn, lớp mới tái định nghĩa để phù hợp với mục đích sử dụng. |
| Lớp con kế thừa được những tài sản nào(thuộc tính, phương thức) của lớp cha ?   * Pulic hoặc protected của lớp cha. Trường hợp default thì cha và con phải dùng trong package. |
| Lớp Object là gì   * Là lớp cha của mọi class trong java |
| Khái niệm đa hình   * Là khả năng đối tượng thể hiện hành vi theo nhiều cách khác nhau tùy thuộc vào ngữ cảnh * Để thu được tính đa hình thì java dùng 2 cơ chế overloading và overriding |
| Phân biệt Overloading và Overriding   |  |  | | --- | --- | | Overloading (nạp chồng) | Overriding (ghi đè) | | * Là cơ chế cho phép thực hiện bên trong 1 class. * Có thể định nghĩa ra nhiều phương thức có thể cùng tên nhưng khác nhau tham số truyền vào * Đa hình tại compile | * Là cơ chế cho phéo lớp con định nghĩa lại phương thức đã được định nghĩa trước đó của lớp cha. * Phương thức override phải cùng tên , cùng danh sách tham sô, cùng kiểu trả về * Access modifier phải có level bằng hoặc cao hơn so với phương thức của lớp cha * Đa hình tại runtime | |
| ép kiểu là gì ? các loại ép kiểu   * Ép kiểu là việc gán giá trị của 1 biến có KDL này sang 1 biến có KDL khác * Có 2 loại ép kiểu: * Nới rộng : là kiểu từ KDL có kích thước nhỏ hơn sang KDL có kích thước lớn hơn. Kiểu này không làm mất thông tin * Thu hẹp : là kiểu từ KDL có kích thước lớn hơn sang KDL có kích thước nhỏ hơn. Kiểu này có thể làm mất thông tin |
| Tính trừu tượng là gì ?   * Là khả năng ẩn các chi tiết của trình triển khai , chỉ hiển thị tính năng/ kết quả cho người dùng🡪chỉ quan tâm đến kết quả đạt được, không quan tâm đến cách thực thi/implement * Trong java tính trừu tượng được thể hiện qua abstract class và interface |
| Phân biệt **abstract class** và **abstract interface**   |  |  | | --- | --- | | **abstract class** | **abstract interface** | | Abstract class cũng là 1 lớp có những đặc điểm của 1 lớp bình thường : phương thức, thuộc tính, contructor nhưng contructor  không thể tạo được đối tượng.   * Lớp trừu tượng nhất định phải có các method abstract và non abstract thì có thể có hoặc không tùy * Lớp trừu tượng không hỗ trợ đa kế thừa(chỉ kế thừa được 1 lớp abstract class) * Lớp trừu tượng có thể có biến final , non final, static, non static * Sử dụng: extends * Đại diện mqh: is-a(là) | Interface là 1 bản hợp đồng có những điều khoản in sẵn khi kí hợp đồng thì bắt buộc phải dùng hết tất cả các điều khoản đó.  Interface có 100% là abstract nên không thể khởi tạo đối tượng interface   * Interface chỉ có thể phương thức abstract * Interface hỗ trợ đa kế thừa(có thể kế thừa 1 hoặc nhiều interface khác) * Interface chỉ có biến final, static * Sử dụng: implements * Đại diện mqh: has-a(có thể) | |
| So sánh Array và ArrayList   |  |  | | --- | --- | | Array | ArrayList | | * Có độ dài kích thước cố định * Kiểu dữ kiệu nguyên thủy & đối tượng * Chỉ có thuộc tính length * Tốc độ lưu trữ thao tác nhanh hơn | * Kích thước có thể thay đổi được * Chỉ lưu trữ được kiểu đối tượng( Với kiểu nguyên thủy sẽ được tự động chuyển qua kiểu đối tượng nhờ cơ chế auto-boxing * Có nhiều phương thức * Tốc độ lưu trữ thao tác chậm hơn | |
| So sánh ArrayList và LinkedList   |  |  | | --- | --- | | ArrayList | LinkedList | | * ArrayList dùng mảng động để lưu trữ các phần tử * Thao tác với ArrayList chậm vì nó dùng mảng động nên nếu phần tử nào xóa khỏi mảng, tất cả các phần tử được chuyển trong bộ nhớ | * LinkedList dùng liên kết đôi để lưu trữ các phần tử * Thao tác LinkedList nhanh hơn vì nó sử dụng liên kết đôi nên không cần chuyển đổi các phần tử tron bộ nhớ | |
| Set là gì, các lớp triển khai của Set   * Set là tập hợp các key, không duy trì thứ tự truyền vào, key không trùng nhau, không có null(đại diện là treeSet vì treeset là sắp xếp theo thứ tự tăng dần) * Các lớp triển khai: HashSet, LinkedHashSet, TreeSet |
| Generic là gì   * Là cơ chế kỹ thuật tham số hóa KDL |
| Ưu điểm và hạn chế khi dùng generic ?   * Ưu điểm: phát hiện lỗi ngay tại thời điểm compile   Không ép kiểu   * Hạn chế: chỉ truyền vào 1 kiểu dữ liệu, khống chế sự linh hoạt khi sử dụng KDL. |
| Stack là gì, các phương thức của stack ? |
| Queue là gì, các class triển khai của queue ? |
| Phương thức cơ bản của queue ? |
| So sánh Comparable và Comparator, khi nào dùng cái nào ?   |  |  | | --- | --- | | Comparable | Comparator | | * Implement comparable cho lớp đối tượng cần được so sánh * Comparable cung cấp phương thức compare To() để sắp xếp các phần tử * Cách sử dụng : Collections.sort(List) | * Tạo ra class mới implement comparator * Comparator cung cấp phương thức compare() để sắp xếp các phần tử * Cách sử dụng: Collections.sort(List, Comparator) | |
| Map là gì, các class triển khai |
| Cây nhị phân là gì |
| Trình bày các cách duyệt cây nhị phân. |
| Ngoại lệ (Exception là gì) ? |
| Phân loại Exception |
| Phân biệt Error và Exception |
| Có bao nhiêu cách để xử lý ngoại lệ (handle) |
| Một số lưu ý khi dùng try-catch |