1.Java là gì ? là ngôn ngữ lập trình bậc cao, hướng đối tượng, bảo mật và là 1 platform.

2 JDK, JRE,JVM ?

* JDK ( java development kit ) là bộ công cụ hỗ trợ lập trình viên phát triển java.
* JRE ( java runtime evironment) được sử dụng để cung cấp môi trường thực thi và thư viện để chạy java.
* JVM ( java vitual machine ) cung cấp máy ảo java để thông dịch java byte code ( . class) ra ngôn ngữ máy và hiển thị ra màn hình .

3 Thông dịch , biên dịch ?

* Biên dịch là dịch toàn bộ files 1 lần và sau đó sử dụng lại kết quả mà không cần phải dịch lại. Files .java biên dịch thành files.class
* Thông dịch là dịch từ dòng lệnh files.class ra ngôn ngữ máy và hiển thị ra màn hình. Mỗi lần chạy chương trình là phải dịch lại .

4 Java là ngôn ngữ thông dịch hay biên dịch ?

* Cả 2

5 Các kiểu dữ liệu trong Java và mặc định khi khai báo?

* Kiểu nguyên thủy :
  + Byte : 0
  + Short : 0
  + Int : 0
  + Long : 0L
  + Float : 0.0 f
  + Double : 0.0 d
  + Char : \u0000
  + Boolean : false
* Kiểu đối tượng :
  + String : null
  + Object : null

6 Có bao nhiêu mệnh đề If ?

* If thiếu ( chỉ có If và khối lệnh được thực hiện khi điều kiện đúng )
* If/else : nếu điều kiện đúng thực hiện khối lệnh của if nếu điều kiện sai thực hiện khối lệnh của else
* If lồng nhau : nhiều If được lồng và nhau . được sử dụng khi muốn kiểm tra nhiều điều kiện trước khi thực hiện 1 khối lệnh.
* If bậc thang : nhiều lệnh if/else if liên tiếp nhau .

7 So sánh if và switch-case

|  |  |
| --- | --- |
| If | Switch-case |
| * So sánh > , < , == , khác .. * Mỗi if có 1 biểu thức điều kiện * Biểu thức điều kiện trả về true/false * Mỗi biểu thức điều kiện của if đúng chỉ thực hiện 1 khối lệnh của if đó | * Chỉ so sánh == hoặc khác * Tất cả các case được so sánh với 1 biểu thức điều kiện duy nhất * Biểu thức điều kiện trả về char , int , byte , short hoặc String * Nếu không có break; , thì tất cả các case sau case đúng đều được thực hiện |

8 Khi nào dùng for , while , do-while ?

* For sử dụng khi biết trước số vòng lặp.
* While sử dụng khi không biết trước số vòng lặp
* Do-while sử dụng khi không biết trước số vòng lặp và muốn thực hiện 1 lần trước khi kiểm tra điều kiện .

9 Phân biệt giữa for và for- each ?

|  |  |
| --- | --- |
| for | For each |
| * Có thể duyệt từ đầu tới cuối hoặc ngược lại hoặc từ bất kì vị trí nào trong mảng * Có thể chỉnh sửa mảng | * Chỉ duyệt được từ đầu mảng tới cuối mảng * Không thể chỉ sửa mảng |

10 Phân biệt while và do-while ?

|  |  |
| --- | --- |
| While | Do-while |
| * Đều sử dụng khi không biết trước số lần lặp | |
| * Kiểm tra điều kiện trước rồi mới thực hiện khối lệnh * Điều kiện sai ngay từ lần đầu thì không thực thi khối lệnh | * Thực hiện khối lệnh trước rồi mới kiểm tra điều kiện * Khối lệnh vẫn được thực thi 1 lần dù điều kiện có sai ngay lần đầu tiên |

11 Break và continue có tác dụng gì trong mệnh đề lặp ?

* Break : khi gặp break thì dừng và thoát khỏi vòng lặp ngay lập tức .
* Continue : khi gặp continue thì sẽ bỏ qua những câu lệnh còn lại của vòng lặp hiện tại và sẽ thực hiện những vòng lặp tiếp theo nếu còn .

12 Các cách khởi tạo 1 mảng trong java ?

* DataType [] arrayName = new DataType[ array.length];
* DataType arrayName[] = new DataType[ array.length];
* DataType [] arrayName = {element,…};
* DataType arrayName[] = {element,…};

13 Phần tử của mảng có thể dùng kiểu dữ liệu nào , mặc định là gì ?

* Phần tử của mảng có thể dùng kiểu dữ liệu nguyên thủy hoặc object
* Giá trị mặc định theo giá trị mặc định mà kiểu dữ liệu của mảng khai báo.

14 OOP là gì ?

* OOP : Object oriented programing
* Là ngôn ngữ lập trình hướng đối tượng .

15 Phân biệt Class và Object ?

|  |  |
| --- | --- |
| Class | Object |
| * Là 1 nhóm các đối tượng có chung thuộc tính và phương thức * Là khuôn mẫu tạo ra các đối tượng | * Là thể hiện cụ thể của Class * Có các thuộc tính và phương thưc đặc trưng |

16 Constructor là gì ?

* Là phương thức đặc biệt để khởi tạo đối tượng .

17 Cách khai báo và đặc điểm Constructor

* Constructor phải trùng tên với Class
* 1 lớp có thể có 1 hoặc nhiều constructor , nhưng phải khác tham số
* Nếu không khai báo constructor java sẽ cung cấp 1 constructor mặc định không có tham số .Khi khai báo 1 constructor có tham số thì constructor được java cung cấp sẽ mất đi.

18 Phân biệt constructor và method

|  |  |
| --- | --- |
| Constructor | Method |
| * Là phương thức đặc biệt để khởi tạo đối tượng * Phải trùng tên lớp * Không có kiểu dữ liệu trả về * Java cung cấp mặc định 1 constructor k tham số khi class không khai báo constructor * Không thể khai báo là static/final | * Thể hiện hành vi hành động của đối tượng * Khác tên hoặc trung với tên lớp * Có kiểu dữ liệu trả về * Không cung cấp * Có thể . |

19 Tính bao đóng là gì . Làm sao để thu được tính bao đóng trong JAVA?

* Tính bao đóng là hạn chế quyền truy cập đến thuộc tính của đối tượng của class .
* Thu được tính bao đóng thông qua khai báo thuộc tính = private .
* Khai báo các phương thức getter/setter cần thiết để cấp quyền truy cập đến thuộc tính cần thiết

20 Tham trị tham chiếu ?

* Tham trị : không thay đổi giá trị ban đầu của biến.
* Tham chiếu : thay đổi giá trị ban đầu của biến .

21 Từ khóa static để làm gì .?

Từ khóa static dùng để khai báo thành phần thuộc lớp , quản lí bộ nhớ .

* Biến static ( biến tĩnh ) : sử dụng để tham chiếu đến thuộc tính chung của đối tượng , chỉ lấy bộ nhớ 1 lần tại thời điểm tải lớp , tiết kiệm bộ nhớ
* Phương thức static : áp dụng từ khóa static cho phương thức -> static method
* Khối static : thực thi lúc tải lớp .

22 Ràng buộc khi sử dụng static

* + Thuộc lớp chứ không thuộc đối tượng
  + Có thể gọi mà không cần tạo đối tượng
  + Có thể dùng để thay đổi giá trị static
  + Phương thức static : không thể gọi phương thức non-static , chỉ có thể truy xuất biến static , không thể dùng super, this.

23 Các loại biến trong java

* Biến toàn cục ( instance)
* Biến cục bộ ( local)
* Biến static

24 Access Modifier và phạm vi truy cập :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Class | Package | Subclass | Public |
| Private | X |  |  |  |
| Default | X | X |  |  |
| Protected | X | X | X |  |
| Public | X | X | X | X |

25 Kế thừa (inheritance) trong java ?

* Là cơ chế cho phép lớp con (subclass) sử dụng lại các thuộc tính và phương thức đã được định nghĩa ở lớp cha.Đặc điểm :
  + Không có đa kế thừa
  + Không kế thừa đc các thành phần được khai báo = private và constructor
  + Lớp con có thể gọi được constructor của lớp cha = Super();
  + Lớp con có thể mở rộng các thuộc tính và phương thức khác .

26 Lớp con kế thừa đc tài sản nào của lớp cha ?

* Lớp con kế thừa đc tất cả ngoại trừ các thuộc tính , phương thức được khai báo = private và constructor của lớp cha.

27 Lớp Object là gì ?

* Là lớp cha của tất cả các lớp trong java . là lớp cao nhất trong java

28 Khái niệm đa hình

* Cùng 1 phương thức nhưng mỗi trường hợp cho ra 1 kết quả khác nhau .

29 Phân biệt overloading và overriding

|  |  |
| --- | --- |
| Overloading | Overriding |
| * Xảy ra trên cùng 1 class * Cùng tên nhưng khác tham số * Có thể giống hoặc khác kiểu dữ liệu trả về * Thể hiện tính đa hình lúc biên dịch * Không quan trọng access modifier | * Xảy ra ở 2 class có quan hệ kế thừa * Cùng tên cùng tham số * Phải giống kiểu dữ liệu trả về * Thể hiện tính đa hình lúc runtime * Access modifier của lớp con phải có level >= lớp cha |

30 Ép kiểu là gì ? các loại ép kiểu ?

* Ép kiểu là gán giá trị của 1 biến có kiểu dữ liệu này cho 1 biến có kiểu dữ liệu khác
* Có 2 loại ép kiểu :
  + Ép kiểu ngầm định là ép kiểu ngầm trong chương trình để nới rộng bộ nhớ. Không gây ra mất dữ liệu
  + Ép kiểu tường minh là ép kiểu thủ công để thu hẹp bộ nhớ dễ gây ra mất mát dữ liệu.

31 Tính trừu tượng là gì ?

* Là chỉ quan tâm đầu vào và đâu ra . không quan tâm đến những công việc được thực hiện bên trong .

32 Phân biệt abstract class và abstract interface?

|  |  |
| --- | --- |
| Abstract class | Abstract interface |
| * Đều không thể tạo đối tượng * Đều có tính trừu tượng và đa hình | |
| * Có constructor * Có thể chưa các propety và method thường * 1 lớp thường chỉ có thể extends 1 abstract class * Access modifier tùy chỉnh nhưng không thể khai báo final * Dùng khi xác định mối quan hệ là is -a | * Không có constructor * Chỉ chưa hằng số và abstract method * 1 lớp thường có thể implements 1 hoặc nhiều interface * Access modifier mặc định là public ( với hằng số là public static) * Dùng khi xác định mối quan hệ là has-a hoặc can-do |

33 So sánh array và arrayList

|  |  |
| --- | --- |
| Array | ArrayList |
| * Không thể co giãn * Có thể chứa 2 kiểu dữ liệu nguyên thủy và đối tượng * Chỉ hỗ trợ array.length | * Có thể co giãn * Chỉ có thể chứa kiểu đối tượng ( muốn chứa kiểu dữ liệu nguyên thủy thì Java cung cấp cơ chế auto-boxing để chuyển từ kiểu dữ liệu nguyên thủy sang wrapper class * Hỗ trợ nhiều phương thức để thao tác |

34 So sánh ArrayList và LinkedList

|  |  |
| --- | --- |
| ArrayList | LinkedList |
| * Đều duy trì thứ tự phần tử thêm vào * Đều không đồng bộ | |
| * Lưu phần tử trong mảng động * Truy xuất và sửa nhanh * Thêm và xóa chậm | * Sử dụng danh sách liên kết để lưu phần tử ( các phần tử được lưu dưới dạng các node) * Truy xuất và sửa chậm * Thêm và xóa nhanh |

35 Set là gì ? Các lớp triển khai của set ?

* Set là 1 interface kế thừa collection interface ,Trong Java set không thể chứa các phần tử trùng lặp .
* Các lớp triển khai :
  + HashSet : lưu trữ các phần tử trong bảng băm , không bảo đảm thứ tự lưu của các phần tử khi chèn vào
  + TreeSet : lưu trữ phần tử trong 1 cây , sắp xếp các phần tử dựa trên giá trị của chúng
  + LinkHashSet : lưu trữ các phần tử trong bảng băm với cấu trúc dữ liệu liên kết . sắp xếp các phần tử của nó theo thứ tự chèn vào
  + EnumSet là cài đặt chuyên biệt dành cho kiểu enum.

36 Generic là gì ?

* Là cơ chế cho phép sử dụng kiểu dữ liệu như 1 tham số để xác định các kiểu dữ liệu cụ thể cho các lớp , phương thức trong các trương hợp khác nhau.

37 Ưu điểm , hạn chế của generic ?

* Kiểu dữ liệu an toàn
* Không phải ép kiểu
* Kiểm tra lúc biên dịch nên dễ nhận biết lỗi sai khi viết code
* Không thể gọi generic = kiểu dữ liệu nguyên thủy thay vào đó java cung cấp cơ chế auto-boxng để chuyển từ kiểu nguyên thủy sang kiểu object
* Không thể tạo đối tượng generic , không thể sử dụng static cho generic, không thể ghi chồng phương thức generic

38 Stack là gì ? Phương thức của stack?

* Strack ( ngăn xếp ) : lưu trữ các phần tử dưới dạng ngăn xếp theo cơ chết last in first out
* Các phương thức :
  + Push() : thêm phần tử vào cuối ngăn xếp
  + Pop() : xóa phần tử ở cuối ngăn xếp
  + isEmpty() : kiểm tra ngăn xếp có rỗng
  + peek(): lấy phần tử cuối ra mà k xóa khỏi stack
  + isFull() kiểm tra ngăn xếp có đầy chưa

39 Queue là gì? Các class triển khai của Queue ?

* Queue : lưu trữ các phần tử dưới dạng hàng dợi theo cơ chết first in first out
* Các class triển khai :
  + ArrayDeque
  + LinkedList
  + PriorityQueue

40 Các phương thức cơ bản trong Queue?

* Add() : thêm 1 phần tử vào queue . nếu đúng trả về true nếu sai ném ra 1 ngoại lệ
* Offer(): thêm 1 phần tử vào queue , nếu đúng trả về true nếu sai trả về false
* Element() trả về dầu hàng đợi , nếu hàng đợi trống ném ra 1 ngoại lệ
* Peek() trả về đầu hàng đợi nếu hàng đợi trống trả về null
* Remove() trả về và xóa phần tử đầu của hàng đợi , ném ra 1 ngoại lệ nếu hàng đợi trống
* Poll() :trả về và xóa phần tử đầu của hàng đợi , trả về null nếu trống

41 So sánh comparable và comparator, khi nào dùng cái nào ?

|  |  |
| --- | --- |
| Comparable | Comparator |
| * Đều sử dụng để sắp xếp các phần tử trong mảng , list | |
| * Phải implements Comparable Interface cho lớp đối tượng cần được so sánh * Cung cấp phương thức compareTo() để sắp xếp * Thuộc java.lang package * Chỉ có thể so sánh 1 tiêu chí mỗi lần thay đổi tiêu chí phải sửa trong source nguồn . dễ gây ra bug | * Không cần implements Comparator cho lớp đối tượng cần được so sánh * Cung cấp phương thức compare() để sắp xếp * Thuộc java.util package * Có thể so sánh nhiều tiêu chí mà không cần sửa source nguồn |

42 Map là gì ? Các lớp triển khai?

* Map sử dụng để lưu trữ và truy xuất dữ liệu theo cặp key-value. Đặc điểm :
  + Các key là duy nhất
  + Chỉ có thể có 1 key là null nhưng có thể có nhiều giá trị null
  + 1 key ứng với 1 value
* Các lớp triển khai :
  + HashMap : giống map nhưng lưu phần tử không duy trì theo thứ tự thêm vào.
  + LinkedHashMap : giống map nhưng duy trì thứ tự thêm vào
  + TreeMap : giống map nhưng không cho phép key là null , và sắp xếp theo thứ tự tăng dần theo key

43 Cây nhị phân là gì ?

* Là cấu trúc dữ liệu phân cấp theo mối quan hệ cha – con mà với mỗi node cha thì chỉ có tối đa 2 node con

44 Trình bày cách duyệt cây nhị phân ?

* Duyệt tiền thứ tư : Duyệt nút gốc đầu tiên . nếu nhỏ hơn nút gốc thì duyêt cây con bên trái , nếu lớn hơn nút gốc duyệt cây con bên phải
* Duyệt trung thứ tự : Duyệt bên trái nút gốc -> duyệt nút gốc -> duyệt bên phải nút gốc
* Duyệt hậu thứ tự : Duyệt cây con bên trái -> duyệt cây con bên phải -> duyệt nút gốc

45 Ngoại lệ ( Exception ) là gì ?

* Là những sự kiện xảy ra có thể phá vỡ luồng thực thi thậm chí là dừng chương trình .

46 Phân loại Exception ?

* Checked Exception là những ngoại lệ xảy ra tại thời điểm biên dịch. Bắt buộc phải xử lí những ngoại lệ này
* Uncheck Exception là những ngoại lệ xảy ra tại thời điểm runtime . Loại này được bỏ qua tại thời điểm biên dịch , Và không bắt buộc chúng ta phải xử lí
* User Exception là những ngoại lệ do người dùng định nghĩa để xử lý 1 số trường hợp cần thiết

47 Phân biệt Error và Exception ?

|  |  |
| --- | --- |
| Exception | Error |
| * Đều là những lỗi xảy ra có thể dẫn đến việc dừng chương trình | |
| * Ngoại lệ có thể được kiểm soát bởi lập trình viên | * Là những vấn đề nghiêm trọng của hệ thống hoặc môi trường thực thi mà lập trình viên không thể kiểm soát |

48 Có bao nhiêu cách để xử lý ngoại lệ ?

* Sử dụng try – catch - finally , throw , throws

49 Một số lưu ý khi dùng try-catch ?

* Sau 1 try phải có 1 catch hoặc 1 finally
* Mỗi try chỉ có 1 finally duy nhất . nhưng có thể có nhiều catch
* Level của catch sau phải lớn hơn hoặc = catch trước
* Finally luôn được thực thi dù có xảy ra ngoại lệ hay không trừ khi thoát chương trình bằng System.exit(); hoặc có 1 error xảy ra làm chết chương trình.
* Tại 1 thời điểm chỉ xảy ra 1 ngoại lệ và chỉ có 1 khối catch ngoại lệ được thực thi , khi exception đã bị catch thì những catch sau sẽ k được bắt.