1 Java là gì ?

- Là ngôn ngữ lập trình hướng đối tượng , Write once, run anywhere ( viết 1 lần chạy nhiều nơi)

2 JDK, JRE, JVM

- JDK là bộ công cụ hỗ trợ lập trình Java

- JRE môi trường thực thi ứng dụng Java

- JVM máy ảo Java – thực thi mã byte – code ( .class ) ra ngôn ngữ máy

3 Thông dịch, biên dịch khác nhau như thế nào ?

- Biên dịch: dịch toàn bộ file 1 lần từ .java -> .class (byte code)

- Thông dịch: dịch từ dòng lệnh từ .class (byte code) -> mã máy

4 Java là ngôn ngữ thông dịch hay biên dịch ?

- Vừa biên dịch vừa thông dịch

5 Java Platform là gì?

- Bao gồm 1 công cụ thực thi, trình biên dịch và một bộ thư viện Java

- Java là ngôn ngữ độc lập với nên tảng. Nó không dành riêng cho bất kỳ bộ xử lý hoặc hệ điều hành nào, có nghĩa chỉ cần cài Java Platform thì bất kỳ xử lý hoặc hệ điều hành nào cũng có thể chạy Java

- Java SE: Java core

- Java EE: phát triển web

- Java ME: phát triển ứng dụng mobile và các thiết bị nhỏ

- Java FX: phát triển ứng dụng liên quan đến mạng

6 Các kiểu dữ liệu trong java, và giá trị mặc định khi khai báo.

- Kiểu nguyên thủy:

+ byte: 1 byte Giá trị mặc định: 0

+ short: 2 byte 0

+ int: 4 byte 0

+ long: 8 byte 0L

+ float: 4 byte (vô hạn) 0.0f

+ double: 8 byte (vô hạn) 0.0d

+ boolean: 1 bit (0: false ; 1: true) false

+ char: 2 byte (kí tự) \u0000

- Kiểu đối tượng: String, Object, Array

7 Sự khác nhau giữa bộ nhớ heap và stack trong java?

- Bộ nhớ stack được sử dụng để lưu trữ các biến cục bộ và gọi hàm trong khi bộ nhớ heap được sử dụng để lưu trữ các đối tượng trong Java

8 Trình bày các cách khởi tạo một mảng trong JAVA ?

- Dùng từ khóa new:

+ ElementType []array\_name/ array\_name[] = new int[5]; => {0, 0, 0, 0, 0}

- Khởi tạo nhanh bằng cách gán trực tiếp giá trị có sẵn ban đầu:

+ ElementType []array\_name/ array\_name[] = {1, 2, 3, 4, 5}

9 Phần tử của mảng có thể dùng kiểu dữ liệu nào, và có giá trị mặc định là gì ?

- Phần tử của mảng phải có cùng kiểu dữ liệu

- Giá trị mặc định là null

10 OOP là gì ?

- Object oriented programming là kỹ thuật lập trình giúp ánh xạ các đối tượng có trong thực tế vào trong lập trình

11 Phân biệt class và object

|  |  |
| --- | --- |
| **Class** | **Object** |
| - Khai báo class dùng từ class | - Đối tượng được khởi tạo bằng từ khóa new |
| - Chỉ được khai báo 1 lần trong cùng 1 packge | - Đối tượng có thể được khởi tạo nhiều lần |

12 Constructor là gì

- Là phương thức đặc biệt dùng để khởi tạo đối tượng

13 Cách khai báo constructor và đặc điểm constructor trong Java

- Tên constructor phải trùng với tên Class

- Khi không khai báo constructor thì java sẽ cung cấp 1 constructor mặc định không tham số

- Nếu có khai báo bất kỳ 1 constructor nào thì constructor mặc định không tham số sẽ bị mất đi

- Không có kiểu dữ liệu trả về

- Trong cùng 1 class thì có thể có nhiều constructor

- Có thể gọi 1 và chỉ 1 constructor khác bằng từ khóa this và lệnh nằm ở vị trí đầu tiên của body

14 Phân biệt constructor và method

|  |  |
| --- | --- |
| Constructor | Method |
| - Để khởi tạo một đối tượng, tên constructor phải trùng tên Class | - Thực thi các câu lệnh thực hiện 1 nhiệm vụ nhất định |
| - Không có dữ liệu trả về | - Có hoặc không dữ liệu trả về |
| - Không đi cùng với các từ khóa abstract, final, static | - Đi cùng với các từ khóa abstract, final, static |

15 Tính bao đóng là gì ?Làm sao để thu được tính bao đóng trong Java ?

- Là kỹ thuật che giấu thông tin của đối tượng, chỉ hiển thị những thông tin cần thiết. Thuộc tính tuyệt đối không được để public => nên dùng private

- Để đạt được tính bao đóng thì trong java dùng: Access modifier, getter, setter

16 Tham trị, tham chiếu

- Tham trị: sao chép dữ liệu vào hàm => đối số không bị thay đổi khi đi qua hàm(trường dữ liệu truyền vào là kiểu nguyên thủy)

- Tham chiếu: sao chép địa chỉ tham chiếu vào hàm => đối số có thể bị thay đổi khi qua hàm (trường hợp dữ liệu là kiểu đối tượng)

17 Từ khóa static dùng để làm gì ?

- Là từ khóa dùng để khai báo thuộc tính properties, method (block, class) của class không phải của đối tượng

- Truy suất thành phần static thông qua tên class hoặc có thể thông qua đối tượng

- Có thể truy suất các thành phần static mà không cần khởi tạo đối tượng

- Mục đích:

+ Định nghĩa chung các thuộc tính, phương thức chung cho toàn bộ đối tượng

+ Tạo ra các lớp tiện ích

18 Ràng buộc khi sử dụng static

- Phương thức static chỉ có thể truy cập phương thức static khác

- Phương thức static chỉ có thể truy cập được các biến static

- Muốn khởi tạo giá trị cho biến static thì có thể gán trực tiếp hoặc khởi tạo thông qua khối static

19 Các loại biến trong Java

- Biến instance: khai báo trong 1 class, bên ngoài constructor, method và block {}

- Biến local: khai báo bên trong constructor, method hoặc block {}

- Biến static: khai báo bằng từ khóa static

20 Trình bày các loại access modifier, và phạm vi truy cập.

- Có 4 loại :

+ private : truy cập bên trong lớp

+ default : truy cập bên trong lớp, package

+ protected : truy cập bên trong lớp, package và bên ngoài package chỉ bởi lớp con

+ public : truy cập công khai, tự do từ bên ngoài

21 Kế thừa trong Java là gì ?

- Là cơ chế cho phép lớp con sử dụng lại đặc điểm và hành vi đã được định nghĩa ở lớp cha

22 Lớp con kế thừa được những tài sản nào(thuộc tính, phương thức) của lớp cha ?

- Lớp con kế thừa hết tất cả các thuộc tính và phương thức của lớp cha **TRỪ** những trường hợp sau:

+ Lớp con không thể kế thừa constructor của lớp cha

+ Lớp con không thể truy cập tài sản private của lớp cha

+ Java không hỗ trợ đa kế thừa (đối với class)

23 Lớp Object là gì

- Là loại lớp gốc của tất cả các lớp trong Java

+ toString(): phương thức trả về chuỗi biểu diễn của đối tượng

24 Khái niệm đa hình

- Là khả năng 1 đối tượng thể hiện hành vi theo nhiều các khác nhau tùy thuộc vào ngữ cảnh

25 Phân biệt Overloading và Overriding

|  |  |
| --- | --- |
| Override (ghi đè) | Overload (nạp chồng) |
| - Là cơ chế cho phép lớp con định nghĩa lại nội dung các phương thức đó ở lớp cha | - Là cơ chế cho phép 1 lớp có khả năng định nghĩa ra nhiều phương thức cùng tên nhưng khác nhau về tham số truyền vào |
| - Phương thức override phải cùng tên cùng danh sách tham số, cùng kiểu trả về | - Xảy ra trong cùng 1 class |
| - Access modifier của lớp con phải có level bằng hoặc cao hơn của lớp cha (ngoại trừ private) | - Đa hình tại compile |
| - Xảy ra trong class có mối quan hệ is a (là một) |  |
| - Đa hình tại runtime |  |

26 Ép kiểu là gì ? các loại ép kiểu

- Ép kiểu ngầm định: diễn ra 1 cách tự động bởi hệ thống (từ con sang cha)

+ int a = 5;

long b = a;

- Ép kiểu tường minh: không thể ép kiểu tự động được, coder phải chỉ rõ kiểu dữ liệu cần ép (từ cha sang con)

+ long a = 5;

int b = (int) a;

27 Tính trừu tượng là gì ?

- Là khả năng loại bỏ đi các chi tiết của quá trình triển khai, chỉ hiển thị tính năng/ kết quả cho người dùng

- Thể hiện tính trừu tượng trong java là abstract method (là 1 method khai báo không có thân hàm)

28 Phân biệt abstract class và interface

|  |  |
| --- | --- |
| Abstract Class | Interface |
| - Là class có tính trừu tượng cao đến mức không thể khởi tạo đối tượng | - Là một bản thiết kế của một lớp, quy định các hành vi chung cho lớp triển khai nó (dùng từ khóa implements) |
| - Không khởi tạo đối tượng nhưng vẫn cung cấp constructor | - Các method toàn bộ là abstract |
| - Lớp abstract có thể chứa thuộc tính và phương thức bình thường | - Không thể khởi tạo đối tượng |
| - Một lớp chứa method abstract thì bắt buộc lớp đó phải là lớp abstract ngược lại thì không | - Không tạo được constructor |
| - Không dùng final cho lớp abstract vì lớp abstract dùng để mở rộng và kế thừa | - Các trường trong interface đều là hằng số |
| - Lớp abstract có thể extends từ lớp abstract khác (không cần override method abstract lớp cha) | - Có thể extends từ 1 hoặc nhiều interface khác |

29 So sánh Array và ArrayList

|  |  |
| --- | --- |
| Array | ArrayList |
| - Kích thước cố định | - Kích thước có thể thay đổi |
| - Chỉ có thuộc tính length | - Có nhiều phương thức hỗ trợ |
| - Có thể lưu nguyên thủy lẫn đối tượng | - Chỉ cho phép lưu kiểu đối tượng (muốn sử dụng kiểu nguyên thủy thì phải sử dụng wapper class) |
| - Tốc độ thao tác của Array nhanh hơn | - Tốc độ lưu trữ và thao tác sẽ chậm hơn |

30 So sánh ArrayList và LinkedList

|  |  |
| --- | --- |
| ArrayList | LinkedList |
| - Sử dụng mảng để lưu trữ | - Sử dụng danh sách liên kết để lưu trữ |
| - Thêm và xóa chậm hơn | - Thêm và xóa nhanh hơn |
| - Tốc độ truy xuất nhanh hơn | - Tốc độ truy suất sẽ chậm hơn |

31 So sánh List, Set và Queue ?

- List: là cấu trúc dữ liệu máy tính -> lưu theo thứ tự nhất định cho phép các phần tử trùng lặp

- Set: mỗi phần tử chỉ xuất hiện 1 lần

- Queue: hàng đợi hoạt động theo cơ chế vào trước ra trước FIFO (first in first out)

32 Set là gì, các lớp triển khai của Set

- Set cho phép lưu các phần tử không trùng lặp

- Có 3 lớp triển khai: hashSet, linkedHashSet, treeSet

+ hashSet: các phần lưu trữ dưới dạng mã băm (hash code), không duy trì thứ tự chèn vào

+ linkedHashSet: các phần tử vẫn lưu dưới dạng mã băm nhưng duy trì thứ tự chèn vào (do sử dụng linked)

+ treeSet: phần tử được sắp xếp (mặc định là sắp xếp tăng dần)

33 Generic là gì

- Là cơ chế cho phép sử dụng kiểu dữ liệu như là tham số => cho phép chúng ta tạo ra các class, method, interface hoạt động với nhiều kiểu dữ liệu khác nhau

34 Ưu điểm và hạn chế khi dùng generic ?

- Xây dựng các thuật toán tổng quát => tái sử dụng mã nguồn

- Quy tắc đặt tên: E - element, N – number, T – type, K - key, V - value

35 Stack là gì, các phương thức của stack ?

- Ngăn Xếp là 1 cấu trúc dữ liệu thêm và lấy phần tử theo nguyên tắt FILO(FOLI)

+ push(): thêm phần tử vào cuối

+ pop(): lấy ra và xoá phần tử

+ peek(): lấy ra xem nhưng ko xoá

+ isEmpty(): kiểm tra mảng rỗng

+ size():

36 Queue là gì, các class triển khai của queue ?

- Là hàng đợi được xây dựng theo nguyên tắt FIFO

+ LinkedList

+ ArrayDeque

+ PriolityQueue

37 Phương thức cơ bản của queue ?

- Add()/Offer: thêm mới

- Remove/poll: lấy và xoá

- Elemement/peek:lấy ra nhưng ko xoá

38 So sánh Comparable và Comparator, khi nào dùng cái nào ?

|  |  |
| --- | --- |
| Comparable | Comparator |
| Phải Implements interface Comparable cho lớp đối tượng cần đươc so sánh | Không phải Implement Comparator cho lớp đối tượng cần được so sánh |
| CompareTo() để sắp xếp các phương thúc | Compare() để sắp xếp các phương thức |
| chỉ được override lại một lần và xét 1 tiêu chí nhất định | compare() có thể được gọi nhiều lần và định nghĩa ngay tại vị trí cần so sánh để sắp xếp theo các tiêu chí tùy ý |
| thuộc về java.lang package | thuộc về java.ultil package |
| có thể sắp xếp các phần tử kiểu Comparable bởi phương thức Collections.sort(List) | có thể sắp xếp các phần tử kiểu Comparator bởi phương thức Collection.sort(List,Comparator) |

39 Map là gì, các class triển khai

- Sử dụng để lưu trữ và truy xuất theo cặp khoá key và value.

+ Mỗi cặp key và value được gọi là entry

+ Map không cho phép 2 key trùng lặpss

+ Mỗi key tương ứng với 1 value

* Các class triển khai:

+ HashMap:

- Lưu trữ dưới dạng cặp key và value

- Nó chứa các key duy nhất

- Chỉ có duy nhất 1 key là null

- Không đảm bảo thứ tự chèn vào

+ LinkedHashMap:

- Giống với hashmap

- Duy trì thứ tự chèn vào

+ TreeMap:

- Giống với map

- Không có key là null

- duy trì thứ tự sắp xếp theo key với 1 thứ tự nhất định (mặc định là thứ tự tăng dần)

+ EnumMap:

40 Phương thức equals() và hashcode() trong Java là gì?

- equals() : trả về boolean

- hashCode() : trả về số nguyên

- Để tối ưu cho việc tìm kiếm, trong Java sử dụng các hash value lấy từ hashcode() của các đối tượng key để gom nhóm các đối tượng này lại thành 1 bucket. Mỗi bucket tương ứng với 1 giá trị hash value

+ Khi tìm kiếm 1 đối tượng thì đầu tiên sẽ lấy hash value để đối tượng key mà chúng ta muốn tìm kiếm sau đó đi kiểm tra các bucket để xem giá trị hash value đó có trùng khớp với bucket hay không. Nếu thoả mãn thì sẽ trả về giá trị của đối tượng value mà chúng ta muốn tìm kiếm.

+ Nhiều đối tượng có thể có cùng hash value do nằm chung 1 bucket, nên Java sẽ sử dụng equals() để tìm chính xác đối tượng value mà muốn tìm kiếm

- Mối quan hệ giữa phương thức equals() và phương thức hashCode() đó là:

+ Nếu hai đối tượng là bằng nhau thì chúng phải có cùng hashCode()

+ Nếu hai đối tượng có cùng hashCode() thì chúng có thể bằng nhau hoặc là không

41 Cây nhị phân là gì

- Tree lưu trữ dữ liệu trên các node

- Các node có mối quan hệ cha con, node trên cùng gọi là node gốc (root node)

- Binary Tree là cây mà mỗi node có 0, 1 hoặc 2 cây con

42 Trình bày các cách duyệt cây nhị phân.

- Có 4 cách duyệt: inorder, postorder, preorder, breath-firt

- Giá trị của node bên trái < node cha

- Giá trị node phải > node cha

43 Ngoại lệ (Exception là gì) ?

- Là 1 sự kiện bất thường xảy ra trong quá trình thực thi 1 ứng dụng => phá vỡ luồng thực thi bình thường của chương trình, thậm chí có thể gây chết chương trình

44 Phân loại Exception

- Checked exception là những ngoại lệ được phát hiện trong quá trình compile. Bắt buộc phải handle

- Unchecked exception là những ngoại lệ được phát hiện trong quá trình runtime. Có thể hoặc không handle cũng được

45 Phân biệt Error và Exception

Error:

- Được định nghĩa trong gói java.lang.Error

- Lỗi xảy ra do thiếu tài nguyên của hệ thống

- Không có cách nào để xử lý lỗi và chương trình sẽ chấm dứt bất thường

- Lỗi được phân loại là loại không được kiểm tra

Exception:

- Được định nghĩ trong gói java.lang.Exception

- Một ngoại lệ xảy ra do vì mã

- Có thể dùng try/catch hoặc throw để xử lý

- Các ngoại lệ được phân loại là loại checked hoặc unchecked.

46 Có bao nhiêu cách để xử lý ngoại lệ (handle)

- Có 2 cách:

+ Try/Catch(finally): dùng để xử lí ngoại lệ ngay tại đoạn mã bị lỗi

+ Thow: ném ngoại lệ cho method khác xử lí

47 Một số lưu ý khi dùng try-catch

- Tại 1 thời điểm chỉ xảy ra 1 ngoại lệ và tại1 thời điểm chỉ có 1 khối catch được thực thi

- Các khối catch phải được sắp xếp từ cụ thể đén diểm chung nhất(exception con->cha)

- Khối finally luôn được thực thiduf chương trình có xảy ra ngoại lệ hay ko

- Khối try có thể có hoặc nhiều khối catch nhưng chỉ có 1 khối finally

- Khối try bắt buộc phải có khối catch hoặc finally 1 trong 2 là được

48 Serialization là gì ? lưu ý khí sử dụng Serialization

- Là cơ chế chuyển đổi trạng thái của 1 đối tượng có 1 chuỗi byte stream. Ngược lại , chuyển đổi từ byte stream về object gọi là Deserialization.

- Trong java để thực hiện cơ chế serialization thì class cần implement interface serializable

- Lưu ý khi sử dụng Sẻialization:

+ Thuộc tính áp dụng từ khoá Static và Transient thì ko thể được Serialization

+ Class cha implement interface Serializable thì class con ko cần implement nữa

49 Phân biệt Throw và Thows

Throw:

- Từ khóa throw trong java được sử dụng để ném ra một ngoại lệ rõ ràng.

- Throw được sử dụng trong phương thức.

- Dùng để trả về một exception được xác định.

- Bạn không thể throw nhiều exceptions.

Throws:

- Từ khóa throws trong java được sử dụng để khai báo một ngoại lệ.

- Throws được khai báo ngay sau dấu đóng ngoặc đơn của phương thức.

- Dùng để thông báo sự tồn tại của exception.

- Có thể khai báo nhiều exceptions, Ví dụ:

public void method()throws IOException,SQLException

50 Phân biệt BufferedReader và FileReader

BufferedReader :

- Sử dụng để đọc các ký tự từ bất kỳ

- Sử dụng bộ đệm bên trong để đọc các ký tự từ

- Tốc độ nhanh hơn và hiệu quả hơn vì nói đọc một dòng

FileReader:

- Dùng để đọc file

- Không sử dụng bộ đệm. Đọc trực tiếp từ tệp bằng cách truy cập ổ cứng.

- Tốc độ chậm hơn và kém hiệu quả hơn vì nói chỉ đọc 1 lần 1 ký tự

51 Try-with-resources là gì?

- Câu lệnh try dùng để khai báo một hoặc nhiều tài nguyên. Một tài nguyên là một đối tượng và phải được đóng sau khi chương trình kết thúc câu lệnh. Nó được khai báo xuất hiện trong ngoặc đơn ngay sau từ khóa try.

52 Làm thế nào để tạo 1 Exception trong Java?

- Tạo một class (tên của class nên kết thúc là từ Exception) extends Exception thuộc java.lang.Exception

- Tạo 1 constructor có tham số String là thông tin chi tiết của Exception

- Trong constructor chỉ cần gọi super (constructor) và truyền tin nhắn