1. Java là gì?

Java là ngôn ngữ lập trình hướng đối tượng. Slogan “Write one, run anywhere”

Viết một lần chạy được mọi nơi.

Sau khi mã nguồn Java được biên dịch thành bytecode.class JVM là một phần mềm có khả năng chạy các chương trình Java trên các nền tảng khác nhau. JVM làm vai trò "dịch" bytecode thành mã máy phù hợp cho hệ điều hành đang sử dụng, cho phép chương trình chạy một cách đáng tin cậy và hiệu quả.

1. JDK, JRE, JVM.

JDK: Bộ dụng cụ hỗ trợ lập trình và chạy ứng dụng java

JRE: Môi trường thực thi ứng dụng java

JVM: Máy ảo java có nhiệm vụ thực thi mã Java byte code (tiệp .class) bằng cách chuyển đổi nó thành mã máy của hệ điều hành.

1. Thông dịch, biên dịch khác nhau như thế nào ?

Thông dịch: dịch từ dòng lệnh, muốn chạy lại thì phải dịch lại lần nữa

Biên dịch: dịch toàn bộ file một lần sau đó sửa dụng kết quả biên dịch mà không cần biên dịch lần nữa

1. JAVA là ngôn ngữ thông dịch hay biên dịch ?

Java là ngôn ngữ lập trình vừa biên dịch và thông dịch tùy thuộc vào trường hợp của nó.

1. Java Platform là gì?

Java Platform là một tập hợp các chương trình giúp phát triển và chạy các chương trình được viết bằng ngôn ngữ lập trình Java.

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

Kiểu dữ liệu nguyên thủy

Số nguyên

Byte

Short

Int

Long: 0L

Số thực

Float: 0.0f

Double: 0.0d

Ký tự

Char: “\u0000”

Giá trị logic

Boolean: false

Tham chiếu

Có sẵn trong java

Scanner

String

Math

Lập trình viên tự định nghĩa

Student, Teacher: null

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

Heap:

Là một khu vực lưu trữ bộ nhớ được chia sẽ giữa các luồng ứng dụng trong Java, để lưu trữ các đối tượng được tạo trong quá trình chạy chương trình

Heap được quản lý bởi bộ thu gom rác để giải phóng bộ nhớ không sử dụng

Khi tạo đối tượng, bộ nhớ được cấp phát trên Heap.

Stack:

Là một khu vực lưu trữ bộ nhớ được chia sẽ để lưu trữ các biến cục bộ và tham số trong các phương thức và các khối lệnh

Stack tự giải phóng bộ nhớ khi phương thức hoàn thành

Khi tạo đối tượng các biến cục bộ và tham số được lưu trên stack

1. Trình bày các cách khởi tạo một mảng trong JAVA ?

Khởi tạo mảng có kích thước cố định

Khởi tạo mảng có giá trị ban đầu

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

Mảng trong java thì có thể dùng kiểu nguyên thủy hoặc đối tượng

Giá trị mặc định của phần tử trong mảng phụ thuộc vào kiểu dữ liệu của phần tử đó

1. OOP là gì ?

Là ngôn ngữ lập trình hướng đối tượng

Là kỹ thuật cho phép lập trình viên ánh xạ những đối tượng bên ngoài thực tế vào lập trình

1. Phân biệt class và object

Class:

* Là một bản thiết kế tạo ra đối tượng
* Class định nghĩa các thuộc tính và phương thức của đối tượng

Object:

* Đối tượng thể hiện cụ thể của class

1. Constructor là gì

Là một phương thức khởi tạo đối tượng

Constructor được gọi tại thời điểm khởi tạo đối tượng nó khởi tạo các giá trị để cung cấp dữ liệu cho đối tượng

1. Cách khai báo constructor và đặc điểm constructor trong JAVA

Tên constructor phải trùng khớp với class không có kiểu trả về và không có giá trị trả về.

Constructor có thể có hoặc không có tham số

Mặc định nếu không khai báo constructor thì java sẽ cung cấp một constructor mặc định không có tham số

Trong 1 class có thể có nhiều constructor

Có thể gọi constructor khác thông qua từ khóa this và lưu ý từ khóa this phải được gọi đầu tiên trong nội dung

1. Phân biệt constructor và method

|  |  |  |
| --- | --- | --- |
|  | Constructor | Method |
| Kiểu trả về | Không có kiểu trả về | Có hoặc không có kiểu trả về |
| Đặt tên | Trùng tên với class | Có thể trùng hoặc không trùng |
| Biên dịch | Nếu không khai báo thì mặc định java sẽ cung cấp constructor rỗng không tham số. | Java không cung cấp method mặc định |
| Số lần gọi | Constructor chỉ được gọi một lần khi khởi tạo đối tượng | Method có thể gọi được nhiều lần |

1. "Tính bao đóng (Encapsulation) là gì ? Làm sao để thu được tính bao đóng trong java ?"

Là kỹ thuật ẩn dấu thông tin của đối tượng, chỉ hiển thị ra những thông tin cần thiết của đối tượng

Hình ảnh về tính bao đóng

Access modifiler: private, public, default,..

Get, set

1. Tham trị, tham chiếu

Biến tham trị

Lưu trực tiếp giá trị của dữ liệu

Được truyền vào phương thức dưới dạng giá trị của biến đó mà không ảnh hưởng đến biến gốc bên ngoài

Biến tham chiếu

Lưu trữ địa chỉ vùng nhớ của dữ liệu

Được truyền vào phương thức dưới dạng địa chỉ của vùng nhớ chứa dữ liệu và thay đổi dữ liệu trong phương thức cũng ảnh hưởng đến biến gốc của giá trị bên ngoài

1. Từ khóa static dùng để làm gì ?

Là từ khóa dùng để khai báo thuộc tính và phương tức thuộc về class chứ không phải của đối tượng, có thể truy cập các phương thức và biến static mà không cần phải khởi tạo đối tượng.

1. Ràng buộc khi sử dụng static

* Phương thức static chỉ có thể gọi phương thức static khác
* Dùng phương thức thì có thể gọi biến static
* Dùng khối static để khởi tạo giá trị cho biến static

1. Các loại biến trong JAVA

Biến local (biến cục bộ)

* Biến khai báo trong method, contructor, block.
* Không dùng được access modifier và cần khởi tạo giá trị mặc định
* Biến local được tạo khi method, contructor, block thực hiện các chức năng và bị hủy khi MT, CTT, BL hoàn thành xong chức năng

Biến instance (Biến toàn cục)

* Biến được khai báo trong lớp, ngoài contructor, method, block
* Biến cục bộ được tạo khi mà đối tượng được tạo bởi từ khóa new và hủy khi đối tượng hủy

Biến static

* Khai báo bằng từ khóa static
* Biến static được tạo khi chương trình bắt đầu chạy và hủy khi chương trình dừng

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

Private: trong class

Default: trong class, trong package, ngoài package nếu class có mối quan hệ kế thừa

Public: Toàn bộ phạm vi truy cập

1. Kế thừa (Inheritance) trong JAVA là gì ?

* Là cho phép lớp con sử dụng lại các đặc điểm và hành vi của lớp cha

Mục đích:

Tái sử dụng mã nguồn

* **Java không hỗ trợ đa kế thừa**

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

* Lớp con có thể kế thừa tất cả các thành phần không phải private của lớp cha
* Có thể định nghĩa lại các phương thức của lớp cha trong lớp con để thay đổi hành vi của chúng

1. Lớp Object là gì

* Lớp Object là lớp gốc của hệ thống phân cấp lớp trong java
* Mọi lớp đều là lớp con của Object
* Lớp Object cung cấp các phương thức dùng chung cho tất cả các đối tượng

1. Khái niệm đa hình (polymorphism)

* Là khả năng 1 đối tượng có thể hiện hành vi theo nhiều cách khác nhau phụ thuộc vào ngữ cảnh

1. Phân biệt overloading và overriding

|  |  |
| --- | --- |
| overloading | overriding |
| Thể hiện đa hình tại compile time | Thể hiện đa hình tại run time |
| Thêm hành vi cho phương thức | Thay đổi hành vi hiện tại của phương thức |
| Có thể khác nhau về số lượng và kiểu dữ liệu của tham số | Số lượng và kiểu dữ liệu của tham số phải giống nhau |
| Xảy ra trong cùng 1 class | Xảy ra ở 2 class có quan hệ kế thừa |

1. ép kiểu là gì ? các loại ép kiểu

Ép kiểu:

* Là quá trình chuyển đổi giá trị của một biến từ kiểu dữ liệu này sang kiểu dữ liệu khác

Ép kiểu ngầm định:

* Đây là loại ép kiểu được thực hiện tự động của java khi các kiểu dữ liệu khác nhau được sử dụng trong 1 biểu thức

Ép kiểu tường minh:

* Đây là loại ép kiểu được thực hiện bằng cách sử dụng toán tử ép kiểu

1. Tính trừu tượng (abstraction) là gì ?

Là khả năng ẩn dấu các chi tiết của trình triển khai chỉ hiển thị tính năng và kết quả cho người dùng

1. Phân biệt abstract class và interface

|  |  |
| --- | --- |
| Abstract class | interface |
| Là một lớp trừu tượng (không thể khởi tạo đối tượng) và được định nghĩa bằng từ khóa “abstract” | Là một tập hợp các phương thức trừu tượng (không có thân) và hằng số, được định nghĩa bằng từ khóa “interface” |
| Có thể chứa các phương thức trừu tượng và phương thức không trừu tượng | Không thể chứa các phương thức có thân hoặc các biến chứa dữ liệu |
| Cho phép định nghĩa các thuộc tính và phương thức bắt buộc được sử dụng bởi tất cả các lớp con | Các lớp con cần triển khai (implement) tất cả các phương thức interface để có thể sử dụng được |

1. So sánh Array và ArrayList

|  |  |
| --- | --- |
| Array | ArrayList |
| Có kích thước cố định | Kích thước có thể thay đổi |
| Có 1 thuộc tính length | Có nhiều thuộc tính |
| Có thể lưu kiểu nguyên thủy và đối tượng | Chi lưu được kiểu đối tượng |
| Tốc độ lưu trữ nhanh | Tốc độ lưu trữ chậm |

1. So sánh array list và linked list

|  |  |
| --- | --- |
| Array list | Linked list |
| Sử dụng mảng động để lưu trữ | Sử dụng liên kết đôi để lưu trữ các phần tử |
| Truy xuất ngẫu nhiên sẽ nhanh hơn | Truy xuất ngẫu nhiên sẽ chậm hơn |

1. So sánh list, set, queue, stack

|  |  |  |  |
| --- | --- | --- | --- |
| List | Set | Queue | Stack |
| Là một tập hợp các phần tử có thứ tự và cho phép chứa các phần tử trùng lặp | Là một tập hợp các phần tử không có thứ tự và không cho phép chứa các phần tử trùng lặp | Là một tập hợp các phần tử có thứ tự và được sử dụng để thực hiện các thao tác “đưa vào hàng đợi” và “lấy ra hàng đợi” | Là một tập hợp các phần tử có thứ tự và được sử dụng để thực hiện các thao tác “đưa vào stack” và “lấy ra stack’ |
| Các phần tử trong List được truy cập thông qua chỉ mục (Index) | Không có chỉ mục để truy cập phần tử Set | Có 2 loại queue:   * Queue thông thường * Dequeue(Double Ended Queue) | Chỉ có thể truy cập các phần tử cuối cùng (đỉnh của Stack) |
| Các lớp cơ sở của List:   * List * Array List * Linked List * Vector | Các lớp cơ sở của Set:   * HashSet * TreeSet * LinkedHasSet | Các lớp cơ sở của Queue:   * priorityQueue * Linked List | Các lớp cơ sở của Stack:   * Stack |

1. Set là gì? Các lớp triển khai của Set.

Set:

* Mỗi phần tử chỉ được xuất hiện duy nhất một lần và tập hợp này chưa được sắp xếp

Các lớp triển khai:

* HashSet: không đảm bảo thứ tự các phần tử
* LinkedHashSet: có thể truy cập các phần tử theo thứ tự chúng được thêm vào
* TreeSet: đảm bảo rằng các phần tử được sắp xếp theo thứ tự tăng dần hoặc giảm dần

1. Generic là gì?

* Là cơ chế cho phép sử dụng biến dữ liệu như là tham số (tham số hóa kiểu dữ liệu)

1. Ưu điểm và hạn chế khi sử dụng Generic.

Ưu điểm:

* Phát hiện lỗi ngay tại thời điểm biên dịch
* Không cần ép kiểu dữ liệu
* Xây dựng các thuật toán tổng quát tái sử dụng mã nguồn

Nhược điểm:

* Có thể làm cho mã nguồn khó hiểu hơn
* Có thể gây ra lỗi trộn lẫn kiểu dữ liệu

1. Stack là gì? Các phương thức của Stack

Stack

* Là một cấu trúc dữ liệu danh sách thêm và lấy ra phần tử theo quy tắc vào trước ra sau (FILO)

Phương thức

* Push: thêm phần tử vào đầu ngăn xếp
* Pop: xóa phần tử trên cùng vào đầu ngăn xếp và trả về phần tử đó
* Peek: trả về một phần tử trên cùng nhưng không xóa phần tử đó ra khỏi ngăn xếp
* Empty: kiểm tra xem ngăn xếp có trống không
* Search: Tìm kiểm phần tử được chỉ định trong ngăn xếp và trả về vị trí của phần tử đó tính từ đỉnh của ngăn xếp. Nếu không tìm thấy phần tử đó thì trả về -1.

1. Queue là gì ? các class triển khai của queue.

Queue:

* Là hàng đợi cấu trúc dữ liệu danh sách thêm và lấy phần tử theo quy tắc (FIFO)

Class triển khai:

* Linked List
* PriortyQueue
* Array De Queue

1. Phương thức cơ bản của queue?

* Add: thêm phần tử vào cuối hàng đợi, nếu không thành công sẽ ném ra một ngoại lệ
* Offer: thêm phần tử vào cuối hàng đợi, trả về true nếu thành công false nếu không
* Remove: lấy phần tử ở đầu hàng đợi và loại bỏ nó, nếu hàng đợi rỗng thì nói ném ra một ngoại lệ
* Poll: lấy phần tử đầu ở hàng đợi và loại bỏ nó, nếu hàng đợi rỗng sẽ ném ra ngoại lệ
* Element: lấy phần tử đầu hàng đợi nhưng không loại bỏ nó, nếu hàng đợi rỗng sẽ ném ra ngoại lệ
* Peek: Lấy một phần tử ở đầu hàng đợi nhưng không loại bỏ nó, trả về null nếu hàng đợi rỗng

1. So sánh Comparable và Comparator, khi nào dùng cái nào ?

|  |  |
| --- | --- |
| Comparable | Comparator |
| Bạn phải implements giao tiếp Comparable cho lớp đối tượng cần so sánh | Bạn KHÔNG phải implements giao tiếp Comparator cho lớp đối tượng cần được so sánh |
| Comparable cung cấp phương thức compareTo() để sắp xếp phần tử | Comparator cung cấp phương thức compare() để sắp xếp phần tử |
| Comparable thuộc về thư viện java.lang | Comparator thuộc về thư viện java.util |
| Chúng ta có thể sắp xếp phần tử của kiểu Comparable bởi phương thức collections.sort(List) | Chúng ta có thể sắp xếp các phần tử của kiểu Comparator bởi phương thức conllections.sort(List, Comparator) |
| Sử dụng: khi muốn thứ tự sắp xếp các đối tượng theo thứ tự mặc định, hoặc khi chỉ muốn sắp xếp theo một tiêu chí duy nhất | Sử dụng: Khi muốn sắp xếp các đối tượng theo một tiêu chí khác với tiêu chí mặc định của chúng, hoặc khi đối tượng không hổ trợ Comparator |

1. Map là gì? Các class triển khai của Map

Map:

* Sử dụng để lưu trữ và truy xuất theo cặp khóa (key) và giá trị (value)
* Mỗi cặp khóa key value được gọi là entry
* Map không cho phép 2 key trùng nhau, mỗi key tương ứng với 1 value

Các class triển khai:

* HasMap

Nó sử dụng bảng băm để lưu trữ các cặp khóa giá trị.

* LinkedHashMap

Là một biến thể của hasMap nhưng duy trì thứ tự của các phần tử dựa trên thứ tự chúng được thêm vào Map. Nó cung cấp một Iterator duyệt qua các phần tử theo thứ tự chúng thêm vào

* TreeMap

Triển khai Map bằng cách sử dụng cây đỏ đen để lưu trữ các phần tử theo thứ tự chúng được thêm vào

* So sánh HasMap, LinkedHashMap, TreeMap

|  |  |  |  |
| --- | --- | --- | --- |
|  | HasMap | LinkedHashMap | TreeMap |
| Thứ tự lưu trữ | Không duy trì thứ tự của các phần tử. Khóa và giá trị được lưu một cách không theo thứ tự cụ thể | Duy trì thứ tự của các phần tử dựa trên thứ tự của chúng được thêm vào Map. Các phần tử sẽ được duyệt qua theo thứ tự chúng được thêm vào | Duy trì các phần tử dựa trên sắp xếp tự nhiên của các khóa. Phần tử sẽ được sắp xếp tăng dần theo khóa. |
| Thời gian truy cập | Nhanh | Nhanh | Chậm |
| Tính đồng bộ | Không đồng bộ. Bạn cần tự đồng bộ hóa nếu sử dụng nó trong môi trường đa luồng. | Không đồng bộ | Không đồng bộ |
| Hiệu suất và sử dụng | Hiệu suất tốt |  |  |

1. Phương thức equals() và hashcode() trong java là gì?

* Equals(): so sánh 2 đối tượng có bằng nhau hay không
* Hashcode(): là phương thức để tính toán mã băm của đối tượng

1. Cây nhị phân là gì?

* Là một cấu trúc dữ liệu trong đó mỗi nút của cây có tối đa 2 nút con, được gọi là nút phải và nút trái.
* Một cây nhị phân có thể rỗng hoặc chỉ có một nút gốc.

1. Trình bày các cánh duyệt cây nhị phân

Duyệt theo thứ tự trước (pre – order)

* In ra giá trị của nút hiện tại
* Duyệt qua cây con bên trái của nút hiện tại
* Duyệt qua cây con bên phải của nút hiện tại

Duyệt theo thứ tự giữa ( in – order)

* Duyệt qua cây con bên trái của nút hiện tại
* In ra giá trị của nút hiện tại
* Duyệt qua cây con bên phải của nút hiện tại.

Duyệt theo thứ tự sau (post – order)

* Duyệt qua cây con bên trái của nút hiện tại
* Duyệt qua cây con bên phải của nút hiện tại
* In ra giá trị của nút hiện tại

1. Ngoại lệ ( exception) là gì?

* Là sự kiện bất thường xảy ra trong quá trình thực thi một chương trình Java nó phá vỡ luồng xử lí của chương trình thậm chí gây chết chương trình

1. Phân loại Exception.

Checked Exception:

* Thường xảy ra bắt lỗi tại compileTime
* Không thể bỏ qua trong lúc compile
* Bắt buộc lập trình viên phải gỡ nó

Ví dụ:

* IOException: được ném ra có lỗi khi chương trình xử lí đọc ghi tệp hoặc xử lý mạng
* ClassNotFoundException: được ném ra khi class không được tìm thấy
* SQLException: ném ra khi có lỗi trong quá trình xử lý cơ sở dữ liệu
* FileNotFoundException: được ném ra khi một chương trình cố gắng mở một file để đọc mà không được tìm thấy

UnCheckd Exception:

* Loại exception xảy ra trong quá trình thực thi chương trình (run time)
* Là loại lỗi không chắc chắn xảy ra và có thể bỏ qua trong quá trình compile
* Lập trình viên không bắt buộc phải gỡ lỗi.

Ví dụ:

* NullPointException: ném ra một tham chiếu đến đối tượng null
* ArrayIndexOutOfBoundsException: ném ra khi một chỉ mục mảng không hợp lệ sử dụng
* ClassCastException: Ném ra một đối tượng không thể ép sang kiểu khác

1. Phân biệt error và exception

Error:

* Là lỗi gây chết chương trình và không thể chạy được

VD:

1. OutOfMemoryError: khi hết bộ nhớ
2. StackOverflowError: khi bị tràn ngăn xếp

Exception:

* Là lỗi có thể xử lý trong quá trình viết code

1. Có bao nhiêu cách xử lý ngoại lệ
2. Sử dụng try-catch

* Đặt mã try lại thời điểm có khả năng xảy ra lỗi exception và bắt ngoại lệ bằng catch tương ứng

1. Ném ngoại lệ (throw)

* Có thể ném ngoại lệ bằng từ khóa throw trong trường hợp không mong muốn hoặc không thể xử lý tại chổ

1. Một số lưu ý khi sử dụng try-catch

* Khối try có thể không có catch
* Tại một thời điểm chỉ xảy ra 1 ngoại lệ và chỉ có 1 khối catch được thực hiện
* Khối catch phải được sắp xếp từ cụ thể đến chung chung nhất