Soạn câu hỏi thi audit

Câu 1: Java là gì?

* Là ngôn ngữ lập trình hướng đối tượng OOP ( Object-Oriented Programming).

Câu 2: JDK, JRE, JWM?

* JDK: là một bộ công cụ hỗ trợ cho lập trình viên phát triễn các ứng dụng java ( javaC.exe, java development kits, java.exe …).
* JRE: là môi trường thực thi các ứng dụng java. ( java runtime environment ).
* JVM: là máy ảo java thực thi mã byte code (.class) ra ngôn ngữ máy ( java virtual machine ).

Câu 3: Thông dịch, biên dịch là gì và khác nhau như thế nào?

* Biên dịch: là dịch toàn bộ ra file 1 lần, sau đó sử dụng kết quả sau biên dịch mà không cần dịch lại.
* Thông dịch: là dịch từng dòng lệnh.

Luồng đi: từ file java biên dịch ra file (.class) -- > từ file (.class) thông dịch ra ngôn ngữ máy.

Sự khác nhau:

* BIên dịch:
  + Ưu điểm: sau khi biên dịch một lần, nó có thể chạy mà không cần biên dịch lại và hiệu quả chạy cao.
  + Nhược điểm: tính di động kém và không đủ linh hoạt.
* Thông dịch
  + Hiệu suất đa nền tảng tốt, thông qua các trình thông dịch khác nhau, diễn giải cùng một mã nguồn thành mã máy dưới nền tảng khác nhau.
  + Nhược điểm: thực thi trong khi chuyển đổi nhờ trình thông dịch, hiểu quả rất thấp ( do đọc từng dòng lệnh ).

Câu 4: Java là ngôn ngữ thông dịch hay biên dịch?

* Java là ngôn ngữ vừa thông dịch và biên dịch, vì trình biên dịch java ( java compile ) chuyển đổi các file code thành java byte-code 🡪 rồi sau đó java virtual machine (jvm) mới thông dịch ra ngôn ngữ máy.

Câu 5: java platform là gì?

* Java platform là một tập hợp các chương trình giúp phát triễn và chạy các chương trình được viết bằng ngôn ngữ java.
* Java platform bao gồm các công cụ thực thi (execution engine) and trình biên dịch (compiler) và một bộ thư viện java.

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

Kiểu nguyên thủy:

* Số nguyên: byte (0: 1byte), short (0: 2byte), int (0: 4byte), long (0L: 8 byte).
* Số thực: double (0.0d: 8 byte), float (0.0f: 4byte).
* Ký tự: char(`\u0000` : 2 byte).
* Luân lý: Boolean (false: 1byte).

Kiểu tham chiếu:

* String and object.

Câu 7: Sự khác nhau giữa bộ nhớ Heap và bộ nhớ Stack trong java.

Bộ nhớ Stack được sử dụng để lưu trữ các biến cục bộ và gọi hàm.

Bộ nhớ Heap được sử dụng để lưu trữ các đối tượng.

Câu 8: Trình bày các cách khởi tạo mảng trong java?

Có 2 cách khởi tạo:

+ Cách 1: int[] array = new int[size];

+ Cách 2: int array[] = {0,1,10,5…};

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

* Nếu là trùng với kiểu dữ liệu nguyên thủy thì giá trị mặc định theo kiểu dữ liệu nguyên thủy.
* Nếu là kiểu đối tượng thì giá trị mặc định là null.

Câu 10: OOP là gì?

* OOP ( Object Oriented-Programming) là một kỹ thuật lập trình cho phép lập trình viên ánh sạ các đối tượng có trong thế giới thực vào bên trong ngôn ngữ máy.

Câu 11:

* Phân biệt class và object

|  |  |
| --- | --- |
| Class | Object |
| + Là khuôn mẫu định nghĩa ra các thuộc tính và hành vi chung của tất cả các đối tượng cùng loại.  + Dùng từ khóa class để khai báo.  + Class chỉ khai báo 1 lần trong phạm vi package. | + đối tượng là thể hiện cụ thể của 1 class.  + dùng từ khóa new để tạo đối tượng  + Đối tượng có thể khởi tạo nhiều lần. |

Câu 12,13: Constructor là gì?

* Constructor là phương thức dùng để khởi tạo đối tượng.

+ Tên constructor trùng với tên class.

+ Không có kiểu trả về.

+ Trong 1 class có thể có nhiều constructor.

+ Mặc định nếu không có constructor thì java sẽ cung cấp 1 constructor ngầm định và có body rỗng.

+ Có thể gọi constructor khác thông qua từ khóa this, this phải để ở đầu trong nội dung của phương thức khởi tạo.

Câu 14: Phân biệt constructor và method trong java.

|  |  |
| --- | --- |
| Constructor | Method |
| + Kiểu trả về: Không có kiểu trả về.  + Đặt tên : tên trùng với tên class.  + Biên dịch: nếu không khai báo, java mặc định cung cấp constructor ngầm định.  + Gọi: chỉ được gọi 1 lần khi new đối tượng | + Kiểu trả về: Có or not  + Đặt tên: theo camel case  + Biên dịch: java không cung cấp mặc định method nào.  + Gọi: Có thể gọi nhiều lần. |

Câu 15: Tính bao đóng là gì? Làm sao để thu được tính bao đóng trong java?

* Là một kỹ thuật để ẩn dấu thông tin của đối tượng, chỉ thể hiện thông tin cần thiết.
* Mục đích:

+ bảo vệ trạng thái bên trong của đối tượng.

+ giảm độ phức tạp của chương trình.

* Để đạt được tính bao đóng thì:

+ Access modifier

+ Getter/Setter

Câu 16: tham trị và tham chiếu?

* Tham chiếu( passed by preference)

+ áp dụng kiểu đối tượng (object,array)

+ giá trị của biến sau khi call method có thể bị thay đổi.

* Tham trị ( passed by value)

+ áp dụng cho kiểu nguyên thủy

+ giá trị trước và sau khi call method không đổi.

Câu 17: Từ khóa static dùng để làm gì?

* Static là khai báo thuộc tính và phương thức cho class, khi đó thuộc tính và phương thức khai báo static sẽ chịu sự quản lý bởi class ( không phải đối tượng ).
* Ứng dụng 1: tạo thuộc tính dùng chung cho nhiều đối tượng trong 1 lớp bất kỳ.
* ứng dụng 2: tạo ra các lớp tiện ích

Câu 18: Rằng buộc khi sử biến static?

* Phương thức static chỉ có thể gọi phương thức static
* Phương thức static chỉ có thể truy cập được biến static.
* Phương thức thường thì có thể gọi biến và cả phương thức static.
* Dùng khối static {} để khởi tạo giá trị mặc định cho biến static.

Câu 19: Các loại biến trong java?

* Có 3 loại biến trong java:

+ biến toàn cục: sử dụng cho cả chương trình

+ biến cục bộ: chỉ sử dụng trong method

+ biến static: sử dụng trong method static và cả method thường và dùng chung trong class.

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

* Các loại access modifier:

+ private: trong class

+ default: trong class, trong package

+ protected: trong class, trong package, ngoài package nếu có quan hệ kế thừa.

+ public: ở đâu cx thấy.

Câu 21: Kế thừa trong java là gì?

* Tính kế thừa là cơ chế cho phép lớp con sử dụng lại các thuộc tính và phương thức đã được định nghĩa ở lớp cha.
* Mục đích: tái sử dụng mã nguồn.

Câu 22: Lớp con kế thừa được tài sản nào ( thuộc tính và phương thức ) của lớp cha?

* Lớp con chỉ được phép sở hữu tài sản và thuộc tính để public, protected, default ( nếu cha và con ở cùng package).
* Lớp con không thể truy cập đến các tài sản private của lớp cha.
* Lớp con không thể thừa kế constructor của lớp cha.
* Mối quan hệ giữa lớp cha và lớp con là mối quan hệ ( is-a ).

Câu 23: Lớp object là gì?

* Lớp object là lớp gốc của tất cả các lớp trong java

Câu 24: Khái niệm đa hình?

* Đa hình là khả năng của 1 đối tượng có thể hiện/ hành vi theo nhiều cách khác nhau, theo nhiều ngữ cảnh.

Câu 25: Phân biệt overriding and overloading?

|  |  |
| --- | --- |
| Overriding ( ghi đè ) | Overloading ( nạp chồng ) |
| + Là cơ chế cho phép lớp con định nghĩa lại nội dung, phương thức có trước đó của lớp cha  + Xảy ra ở mối quan hệ kế thừa  + Phương thức overriding phải cùng tên, cùng danh sách tham số, cùng kiểu trả về ( lưu ý: sẽ có TH khác kiểu )  + Access modifier: ở phương thức lớp con phải có level lớn hơn or bằng so với lớp cha.  + Đa hình tại runtime. | + Là cơ chế cho phép lớp định nghĩa ra nhiều phương thức cùng tên nhưng khác tham số truyền vào.  + Chỉ xảy ra trong cùng 1 class.  + Đa hình tại compile. |

Câu 26: Ép kiểu là gì? Và các loại ép kiểu?

* Ép kiểu là quá trình chuyển đổi kiểu dữ liệu trong java và thường được sử dụng trong trường hợp gán một giá trị vào một biến khi mà kiểu của biến và của giá trị là hoàn toàn khác nhau.
* Có 2 loại ép kiểu:

+ Ép kiểu ngầm định: EX. Int 🡪 double

+ Ép kiểu bắt buộc: EX. Int 🡪 String ( lập trình viên phải chịu trách nhiệm cho kiểu ép này ).

Câu 27: Tính trừu tượng là gì?

* Tính trừu tượng là khả năng ẩn đi các chi tiết trong quá trình triễn khai, chỉ thể hiện tính năng/ kết quả cho người dùng ( chỉ quan tâm kết quả, không quan tâm đến cách thực hiện ).
* Trong java tính trừu tượng được thể hiện qua abtract class and interface.

Câu 28: phân biệt abtract and interface?

|  |  |
| --- | --- |
| Abtract | interface |
| + có constructor  + 1 class chứa 1 phương thức abtract thì class đó phải là class abtract  + không được dùng final trong abtract class  + abtract có thể chứa thuộc tính và method bth ( có thể định nghĩa phần thân phương thức ).  + lớp abtract có thể extend từ 1 lớp abtract khác và không yêu cầu overriding method của lớp abtract cha.  + Không hỗ trợ đa kế thừa.  + không thể tạo đối tượng. | + không có constructor  + Một class có thể thực hiện được nhiều interface ( có thể gọi là đa thừa kế ).  + Không thể định nghĩa phần thân của method.  + Mọi Phương thức, property, đều mặc định là public.  + mọi khai báo biến đều là khai báo hằng  + không thể tạo đối tượng |

Câu 29: So sánh arrayList và array?

|  |  |
| --- | --- |
| Array | ArrayList |
| + kích thước mảng cố định  + Chỉ có thuộc tính length  + Có thể lưu trữ kiểu object và nguyên thủy.  + Tốc độ lưu trữ, thao tác nhanh hơn.  + Không có method hỗ trợ sẵn. | + Kích thước có thể thay đổi được  + Có nhiều phương thức hỗ trợ sẵn  + chỉ lưu trữ được kiểu đối tượng ( mảng tự chuyển kiểu nguyên thủy sang kiểu đối tượng nhờ cơ chế auto-boxing).  + Tốc độ chậm hơn  + Một số method hỗ trợ: add, remove, contain, size, IndexOf, LastIndexOf, replace, get, set,… |

Câu 30: So sánh ArrayList and LinkedList.

|  |  |
| --- | --- |
| ArrayList | LinkedList |
| + Sử dụng mảng động để lưu trữ các phần tử  + Truy suất ngẫu nhiên nhanh hơn ( tốt hơn để lưu trữ và lấy dữ liệu ). | + Sử dụng danh sách liên kết đôi để lưu trữ phần tử.  + Truy suất ngẫu nhiên chậm hơn.  + Nhưng chèn, xóa nhanh hơn. ( hiệu quả cho thao tác ). |

Câu 31: So sánh List, set, Queue trong java?

|  |  |  |
| --- | --- | --- |
| List | Set | Queue |
| + Là cấu trúc dữ liệu tuyền tính ( trùng lặp ), trong đó các phần tử được sắp xếp theo 1 thứ tự xác định. Cho phép các phần tử trùng lặp. | + Mỗi phần tử chỉ được xuất hiện duy nhất 1 lần và chưa được sắp xếp. | + Là hàng đợi, được hoạt động theo cơ chế FIFO. |

Câu 32: Set là gì và các lớp triễn khai của set?

* Set là một interface kế thừa collection interface trong java. Set là một collection không thể chứa các phần tử trùng lặp và các phần tử đó là duy nhất.
* Các lớp triễn khai của Set:

+ HashSet: lưu trữ các phần tử của nó trong bảng băm, là cách thực hiện tốt nhất, tuy nhiên nó không đảm bảo về thứ tự các phần tử được chèn vào.

+ TreeSet: lưu trữ các phần tử của nó trong một cây, sắp xếp các phần tử của nó dựa trên các giá trị chúng, (chậm hơn hashSet).

+ LinkedHashSet: được triễn khai dưới dạng bảng băm có cấu trúc dữ liệu danh sách liên kết, sắp xếp các phần tử của nó dựa trên thứ tự chúng được chèn vào tập hợp.

+ EnumSet: Là cài đặt chuyên biệt để sử dụng với các kiểu enum.

Câu 33: Generic là gì?

* Generic là cơ chế cho phép sử dụng kiểu dữ liệu như thế nào là tham số ( tham số hóa kiểu dữ liệu ).

Câu 34: Ưu điểm và nhược điểm khi dùng generic?

* Ưu điểm:

+ Phát hiện lỗi ngày tại thời điểm biên dịch.

+ Không cần ép kiểu dữ liệu

+ xây dựng các thuật toán tổng quát và tái sử dụng mã nguồn.

* Hạn chế:

+ không thể gọi kiểu nguyên thủy

+ không thể new đối tượng

+ không thể bắt exception

+ Không dùng được static

+ Không thể tạo class

+ Không thể new mảng generic.

Câu 35: Stack là gì? Các phương thức của stack?

* Stack ( ngăn xếp ): là 1 cấu trúc dữ liệu dạng danh sách ( list ), thêm và lấy phần tử theo quy tắc LIFO
* Một số phương thức:

+ push(): thêm

+ peek(): lấy ra xem

+ pop(): lấy ra xem và xóa.

+ isEmpty()/ empty: kiểm tra rỗng.

+ search(): tìm kiếm.

Câu 36, 37: Queue là gì? Các phương thức của Queue?

* Queue ( hàng đợi ): là một cấu trúc dữ liệu danh sách ( queue ), thêm và lấy phần tử theo quy tắc FIFO.
* Có 3 lớp triễn khai:

+ LinkedList

+ Array Deque

+ Priority Queue

* Một số phương thức:

+ add(): thêm ( thành công 🡪 true; thất bại 🡪 quăng ra exception).

+ offer(): thêm ( thành công 🡪 true; thất bại 🡪 false ).

+ element(): lấy ra xem ( thành công 🡪 lấy phần tử đầu hàng đợi, và ném ngoại lệ nếu hàng đợi rỗng).

+ peek(): lấy ra xem ( thành công 🡪 lấy phần tử ở đầu hàng đợi, trả về null nếu hàng đợi rỗng).

+ remove(): xóa ( thành công 🡪 xóa phần tử ở đầu hàng đợi, ném ngoại lệ nếu hàng đợi rỗng).

+ poll(): xóa ( thành công 🡪 lấy phần tử ở đầu hàng đợi và xóa nó, trả về null nếu hàng đợi rỗng).

Câu 38: So sánh comparable and comparator, khi nào dùng cái nào?

|  |  |
| --- | --- |
| Comparable | Comparator |
| + Bạn phải implement giao tiếp Comparable cho lớp đối tượng cần được so sánh.  + Comparable cung cấp phương thức compareTo() để sắp xếp các phần tử.  + Comparable thuộc về java.lang package  + Chúng ta có thể sắp xếp các phần tử của kiểu comparable bởi phương thức Collection.Sort(list). | + Bạn không phải implement giao tiếp comparator cho lớp đối tượng cần được so sánh.  + Comparator cung cấp phương thức compare() để sắp xếp các phần tử.  + Comparator thuộc về java.util package.  + Chúng ta có thể sắp xếp các phần tử của kiểu comparator bởi phương thức Collection.sort(List,comparator). |

Câu 39: Map là gì? Các class triễn khai?

* Map được sử dụng để lưu trữ và truy cập theo cặp khóa ( key ) and giá trị ( value ).
* Mỗi cặp key-value được gọi là entry.
* Map không cho phép 2 key trùng nhau
* Mỗi key tương ứng 1 value.

+ Các lớp triễn khai của map là:

* HashMap:
* Điểm giống: Giống với đặc điểm của map
* Điểm khác:
  + Không đảm bảo thứ tự entry được thêm vào
  + Cho phép 1 key null
  + Cho phép nhiều value có thể null
  + Tốc độ truy suất cao
* LinkedHashMap:
* Điểm giống: Giống với hashmap
  + Duy trì các phần tử - entry theo thứ tự chèn vào.
* TreeMap:
* Điểm giống: Giống với map
* Điểm khác:
  + Không có key mang giá trị null
  + Duy trì các phần tử được thêm vào theo thứ sắp xếp ( mặc định sắp xếp tăng dần ).
* Lưu ý: Với key là kiểu Object do người dùng tự định nghĩa thì cần phải triễn khai interface comparable or comparator.

Câu 40: Phương thức equal(), và hashcode() trong java là gì ?

* Equal(): là phương thức dùng để so sánh 2 đối tượng với nhau.
* Hashcode(): trả về một giá trị số nguyên đại diện cho 1 đối tượng ( mỗi đối tượng sẽ có 1 hashcode khác nhau). Khi 2 object bằng nhau thì hashcode() method của chúng cũng trả về giá trị bằng nhau.

Câu 41: Cây nhị phân là gì?

* Binary search tree: Mỗi cây sẽ có 0,1 hoặc tối đa 2 cây con.
* Tất cả giá trị các node bên cây con trái luôn bé hơn giá trị node gốc
* Tất cả các giá trị các node bên cây con phải luôn lớn hơn giá trị node gốc.

( left < root < right ).

Câu 42: trình bày các cách duyệt cây nhị phân?

* Duyệt tiền tự: ( Root – Left – Right )
* Duyệt trung tự ( Left – Root – Right )
* Duyệt hậu tự ( Left – Right – Root )

Câu 43: Ngoại lệ (exception) là gì?

* Ngoại lệ là sự kiện bất thường xảy ra trong quá trình thực thi chương trình của 1 chương trình java, nó phá vỡ flow ( luồng xử lý ) bình thường của chương trình, thậm chí gây chết ctrinh.

Câu 44: phân loại Exception

* Phân loại exception:

+ Checked exception: loại exception xảy ra trong lúc compile time, không thể bỏ qua trong lúc compile, bắt buộc lập trình viên phải handle nó. ( Ex. IOE exception, FileNotFound exception…

+ Unchecked exception: loại exception xảy ra tại lúc thực thi của chương trình, lỗi này không chắc chắn xảy ra, có thể bỏ qua trong qua trình compile, lập trình viên không nhất thiết phải handle. ( Ex. RunTime exception, NullPointer exception, NumberFormat exception,…)

+ Error: Những lỗi nghiêm trọng, liên quan đến MT thực thi ( JVM ), của ứng dụng hệ thống, lập trình viên, lập trình viên không thể can thiệp, có thể gây die chương trình.

Câu 45,46: Có bao nhiêu cách để xử lý ngoại lệ?

* Cách 1: try-catch-(finally) : xử lý ngoại lệ ngay tại đoạn mã bị lỗi
* Cách 2: throw/ throws : để ném ngoại lệ cho method/ logic khác xử lý.
* Lưu ý: khi dùng try-catch.
  + Khối try có thể có nhiều khối catch or hoặc không cần catch. Nhưng chỉ có 1 finally.
  + Một thời điểm thì xảy ra 1 ngoại lệ và tại 1 thời điểm chỉ có 1 khối catch được thực thi.
  + Khối catch phải được sắp xếp từ cụ thể đến chung nhất ( từ exception con 🡪 exception cha )
  + Khối finally luôn được thực thi dù ctrinh có ngoại lệ or not.

* So sánh Throw and Throws

|  |  |
| --- | --- |
| Throw | Throws |
| + Từ khóa throw trong java được sử dụng để ném 1 ngoại lệ rõ ràng  + Ngoại lệ checked không được truyền ra nếu chỉ sử dụng từ khóa throw  + Sau throw là từ khóa new và tên lỗi  + Throw được sử dụng trong phương thức  + bạn không thể throw nhiều exceptions | + Từ khóa throws trong java được sử dụng để khai báo 1 ngoại lệ  + Ngoại lệ checked được truyền ran gay cả khi chỉ sử dụng từ khóa throws  + Sau throws là một hoặc nhiều class  + Throws được khai báo ngay sau dấu đóng ngoặc đơn của phương thức.  + Bạn có thể khai báo nhiều exceptions ( Ex. Public void method() throws IOE exception, SQLE exception. |

Câu 47: Serialization là gì? Lưu ý khi sử dụng serialization?

* Cơ chế Serialization là cơ chế tuần tự hóa cho phép chuyển trạng thái của 1 đối tượng vào 1 byte tream.
* Ngược lại chuyển từ byte stream về object được gọi là deserialization.
* Lưu ý:
  + 1) thuộc tính áp dụng từ khóa static, transient thì sẽ không được serialization.
  + 2) class cha implement interface Serialization thì các lớp con không cần phải implement lại
  + 3) Thuộc tính của đối tượng phải implement serialization.