**CÂU HỎI AUDIT MODULE 2**

**ADVANCED PROGRAMING WITH JAVA 2.0**

|  |  |
| --- | --- |
| STT | Câu hỏi |
| 1 | JAVA là gì ?:  **Java** là một **một ngôn ngữ lập trình hiện đại, bậc cao, hướng đối tượng, bảo mật và mạnh mẽ.** và là một **Platform**.  **Platform:** Bất cứ môi trường phần cứng hoặc phần mềm nào mà trong đó có một chương trình chạy, thì được hiểu như là một Platform. Với môi trường runtime riêng cho mình (JRE) và API, Java được gọi là Platform.  **"Write Once, Run Anywhere"** - viết một lần chạy nhiều nơi. |
| 2 | **JDK, JRE, JVM:**  -JDK: java development kit: bộ công cụ để hỗ trợ lập trình java.  -JRE: java runtime evironment: môi trường thực thi ứng dụng java.  -JVM: java vitual machine: thực thi java bytecode sang ngôn ngữ máy. |
| 3 | **Thông dịch, biên dịch khác nhau như thế nào ?**  -Biên dịch là dịch toàn bộ file 1 lần duy nhất, sau đó sử dụng kết quả biên dịch mà không cần dịch nữa.  -Thông dịch từng dòng lệnh rồi chạy, muốn chạy phải dịch 1 lần nữa |
| 4 | **JAVA là ngôn ngữ thông dịch hay biên dịch ?**  -Java là loại ngôn ngữ vừa biên dịch vừa thông dịch.  -javac.exe có nhiệm vụ **biên dịch** code java sang bytecode: .java sang .class Sau đó java.exe có nhiệm vụ **thông dịch** bytecode trong class file sang mã máy tùy vào chip/hệ điều hành. JVM |
| 5 | **Các kiểu dữ liệu trong java, và giá trị mặc định khi khai báo.**  Có 2 kiểu dữ liệu trong java: kiểu đối tượng và kiểu nguyên thủy.  + Đối tượng: Array,Class,String,interface.  + Nguyên thủy: byte, short, interger, long, float, double, char, boolean.   | **Kiểu dữ liệu** | **Giá trị mặc định** | **Bộ nhớ** | **Giá trị** | | --- | --- | --- | --- | | Boolean | False | 1 bit | -128 đến127 | | Char | '\u0000' | 2 byte | -32768 đến 32767 | | Byte | 0 | 1 byte | -2mu31 đến 2mu31-1 | | Short | 0 | 2 byte | -2mu63 đến 2mu63-1 | | Int | 0 | 4 byte |  | | Long | 0L | 8 byte |  | | Float | 0.0f | 4 byte |  | | Double | 0.0d | 8 byte |  | |
| 6 | **Có bao nhiêu lại mệnh đề if:**if, if else, if - else if |
| 7 | **So sánh if và switch-case** |
| 8 | **Khi nào dùng for, while, do-while ?** |
| 9 | **Phân biệt for-i và for-each**  -for-index chạy index có thể thay đổi được cách chạy của index.  -for-each không điều chỉnh được chỉ có thể duyệt từ đầu đến cuối. |
| 10 | **Khác nhau giữa while, do-while. Cho ví dụ khi nào dùng ?** |
| 11 | **Break, Continue có tác dụng gì trong mệnh đề lặp ?** |
| 12 | **Trình bày các cách khởi tạo một mảng trong JAVA ?**  **Có 2 cách:**  Thông qua từ khóa new.  Thông qua khởi tạo nhanh |
| 13 | **Phần tử của mảng có thể dùng kiểu dữ liệu nào, và có giá trị mặc định là gì ?**  Array có thể lưu kiểu dữ liệu nguyên thủy hoặc đối tượng.  Giá trị mặc định của phần tử trong mảng sẽ phụ thuộc kiểu dữ liệu của mảng. |
| 14 | **OOP là gì ?**  Là 1 kỹ thuật lập trình. Ánh xạ các đối tượng có trong thực tế vào trong các đối tương trong lập trình.  Lớp là khuôn mẫu định nghĩa các thuộc tính và phương thức chung của các đối tượng cùng loại.  Object là thể hiện của 1 lớp. |
| 15 | **Phân biệt class và object:**   |  |  | | --- | --- | | **Object** | **Class** | | 1 thực thể vật lí | 1 thực thể logic | | Đối tượng tạo ra bằng từ khóa new | Đối tượng tạo ra bằng từ khóa class | | Đối tượng có thể tạo được nhiều lần | Class chỉ được khai báo 1 lần | | Đối tương được cấp bộ nhớ khi nó được tạo ra bằng từ khóa new | Tồn tại khi khai báo | |
| 16 | **Constructor là gì**   1. Là phương thức khởi tạo đối tượng. 2. Tên của contructor trùng với tên của class. 3. Không có kiểu trả về. 4. Trong 1 class có thể có nhiều contructor. Nhưng tối thiểu là 1 contructor rỗng. 5. Mặc định không khai báo contructor thì java sẽ cung cấp 1 contructor không tham số. 6. Có thể gọi contructor khác bằng từ khóa this. |
| 17 | **Cách khai báo constructor và đặc điểm constructor trong JAVA** |
| 18 | **Phân biệt constructor và method**   |  |  | | --- | --- | | **constructor** | **method** | | Băt buộc cùng tên với class | Có thể trùng tên với class hoặc ko | | Không có kiểu trả về | Băt buộc phải định nghĩa kiểu trả về | | Chỉ được gọi khi khởi tạo đối tượng | Có thể gọi nhiều lần | | Nếu không khai báo thì java cung cấp ngầm định contructor không đối số | Java không có cấp method mặc định | |
| 19 | **Tính bao đóng là gì ? Làm sao để thu được tính bao đóng trong java ?**  Tính bao đóng là kỹ thuật ẩn giấu thông tin của đối tượng, chỉ thể hiện những thông tin cần thiết.  -Mục đích:  + Bảo vệ được trạng thái bên trong của đối tượng, không cho phép thay đổi trực tiếp giá trị thuộc tính của đối tượng.  + Giảm độ phức tạp của chương trình.  Để đạt được tính bao đóng thì trong java dùng:  +Access modifier.  +Getter,setter. |
| 20 | **Tham trị, tham chiếu**  1**.Tham trị: là truyền giá trị ( passed by value):**  +Áp dụng cho kiểu dữ liệu nguyên thủy.  +Giá trị của biến trước và sau khi gọi method là không thay đổi.  2.**Tham chiếu: Là truyền tham chiếu đến 1 địa chỉ( pased by reference):**  + Áp dụng cho kiểu dữ liệu đối tượng tham chiếu.  + Giá trị của biến sau khi gọi method có thể bị thay đổi. |
| 21 | **Từ khóa static dùng để làm gì ?**  Từ khóa static là từ khóa khai báo thuộc tính và phương thức của class không phải của đối tượng. |
| 22 | **Ràng buộc khi sử dụng static**   1. Phương thức static chỉ được gọi biến static 2. Phương thức static chỉ có thể truy cập được các biến static 3. Chỉ khởi tạo giá trị biến static thông qua khối static. |
| 23 | **Các loại biến trong JAVA**  Có 3 loại biến trong java: biến toàn cục, biển cục bộ và biến static:  **1.Biến toàn cục/global/instance:**  + Khai báo trong class, bên ngoài contructor, method và trong blog source{}.  + Có giá trị khi khởi tạo đối tượng và mất đi khi đối tượng bị hủy.  **2.Biến cục bộ/local:**  + Khai báo bên trong contructor, method hoặc block sourse{}.  + Được khởi tạo và cấp phát vùng nhớ khi contructor, method hoặc block sourse{} được gọi.  + Biến local sẽ bị hủy khi contructor, method hoặc block sourse{} được thực hiện xong.  + Không thể dùng accessmodifier cho biến local. + Cần khởi tạo giá trị mặc định cho biến local khi sử dụng.  **3.Biến static:**  + Được khai báo bằng từ khóa static.  + Biến của lớp chứ không phải của đối tượng.  + Được tạo khi chương trình chạy và bị hủy khi kết thúc chương trình.  + Truy suất thông qua tên class( hoặc đối tượng) |
| 24 | **Trình bày các loại access modifier, và phạm vi truy cập.**  -Có hai loại Access Modifier trong Java, đó là: Access Modifier và Non-access Modifier.  -Access Modifer trong Java xác định phạm vi có thể truy cập của biến, phương thức, constructor hoặc lớp.  -Trong java, có 4 phạm vi truy cập của Access Modifier như sau:  1.private: trong lớp  2.default: trong lớp, trong package.  3.protected: trong lớp, trong package, ngoài package bởi lớp con.  4.public: : trong lớp, trong package, ngoài package bởi lớp con, ngoài package |
| 25 | **Kế thừa trong JAVA là gì ?(inherritain)**  -Kế thừa là cơ chế cho phép lớp con sử dụng các đặc điểm và hành vi đã được định nghĩa ở lớp cha.  -Mối quan hệ giữa lớp con và lớp cha là is-a.  -Java không hỗ trợ đa kế thừa  -Muốn tạo lớp vô sinh thì dùng final class |
| 26 | **Lớp con kế thừa được những tài sản nào(thuộc tính, phương thức) của lớp cha ?**  -Lớp con không thể kể thừa contructor của lớp cha  -Lớp con không thể truy cập tài sản private của lớp cha |
| 27 | **Lớp Object là gì**  Lớp object là lớp gốc của tất cả các lớp trong java.  toString là phương thức trả về chuổi biểu diễn thông tin của đối tượng. |
| 28 | **Khái niệm đa hình(polumophism)**  Là khả năng 1 đối tượng có thể hiện / hành vi theo nhiều cách khác nhau tùy thuộc vào ngữ cảnh.  Để thực thi tính đa hình thì có 2 cơ chế đó là overloading(nạp chồng) và overrding(ghi đè) |
| 29 | **Phân biệt Overloading và Overriding**   |  |  | | --- | --- | | **Overloading** | **Overriding** | | Là cơ chế cho phép 1 lớp có khả năng định nghĩa ra nhiều phương thức có cùng tên nhưng khác nhau về tham số truyền vào | -Là cơ chế cho phép lớp con định nghĩa lại các phương thức đã được định nghĩa trước đó của lớp cha.  -Phương thức overriding phải cùng tên, cùng tham số, cùng kiểu trả về.  Access modifier phải có lever bằng hoặc cao hơn so với phương thức của lớp cha.  -dùng overriding để ghi đè phương thức | | Xảy ra ở trong 1 class | Xảy ra trong 2 class có mối quan hệ IS-A | | Đa hình tại complie | Đa hình tại runtime | |  |  | |
| 30 | **ép kiểu là gì ? các loại ép kiểu: ép kiểu dữ liệu và ép kiểu đối tượng**  Ép kiểu là việc gán giá trị của 1 biến có kiểu dữ liệu này sang biến có kiểu dữ liệu khác.  Có 2 cách ép kiểu là:  1.ép kiểu ngầm định( implicit casting/widening): Diễn ra một cách tự động, ngầm định bởi hệ thống. Mục đích là mở rộng khả năng lưu trữ mà không làm mất mát thông tin dữ liệu.  2. ép kiểu tường minh(Explicit casting./narowwing): Không thể ép kiểu tự động, coder phải chỉ rõ kiểu dữ liệu cần ép qua cú pháp. Mục đích: thu hẹp lại khả năng lưu trữ, có thể làm mất mát thông tin/giá trị ban đầu.  **Ép kiểu đối tượng:** Ép kiểu ngầm định(upcasting) và Ép kiểu tường minh(Downcasting)   1. Ép kiểu ngầm định từ lớp con sang lớp cha. 2. Ép kiểu tường minh từ lớp cha sang lớp con chuyển 1 đối tượng là thể hiện của 1 lớp cha xuống đối tượng là thể hiện của lớp con. |
| 31 | **Tính trừu tượng là gì ?**  Tính trừu tượng trong Java là tính chất không thể hiện cụ thể mà chỉ nêu tên vấn đề. Đó là một quá trình che giấu các hoạt động bên trong và chỉ hiển thị những tính năng thiết yếu của đối tượng tới người dùng. |
| 32 | **Phân biệt abstract class và abstract interface**  Abstract class và interface đều được sử dụng để có được sự trừu tượng mà ở đó chúng ta có thể khai báo các phương thức trừu tượng.   |  |  | | --- | --- | | **Abstract class** | **Interface** | | 1) Abstract class có phương thức **abstract**(không có thân hàm) và phương thức **non-abstract** (có thân hàm). | Interface chỉ có phương thức **abstract**. Từ java 8, nó có thêm **các phương thức default và static**. | | 2) Abstract class **không hỗ trợ đa kế thừa**. | Interface **có hỗ trợ đa kế thừa** | | 3) Abstract class có các biến **final, non-final, static and non-static**. | Interface chỉ có các biến **static và final**. | | 4) Abstract class **có thể cung cấp nội dung cài đặt cho phương thức của interface**. | Interface **không thể cung cấp nội dung cài đặt cho phương thức của abstract class**. | | 5) Từ khóa **abstract** được sử dụng để khai báo abstract class. | Từ khóa **interface** được sử dụng để khai báo interface. | | 6) Ví dụ: public abstract class Shape { public abstract void draw(); } | Ví dụ: public interface Drawable { void draw(); } | |
| 33 | **So sánh Array và ArrayList**   |  |  | | --- | --- | | Array | ArrayList | | Kích thước cố định | Kích thước có thể thay đổi được | | Có thể lưu trữ dữ liệu kiểu nguyên thủy và đối tượng | Chỉ có thể lưu trữ kiểu đối tượng, nhưng kiểu nguyên thủy có thể tự chuyển đổi gọi là auto-boxing | | Tốc độ lưu trữ và thao tác nhanh hơn | Tốc độ lưu trữ và thao tác chậm hơn | | Chỉ có thuộc tính length() | Có nhiều phương thức để thao tác với dữ liệu |   **Các phương thức của ArrayList:**  -get(): lấy về 1 phần tử.  -add() thêm 1 phần tử.  -remove() xóa phần tử.  -size(): lấy về số lượng phần tử.  -find() tìm kiếm phần tử.  -isEmpty(): kiểm tra rỗng.  -indexOf() Lấy chỉ số của phần tử, nếu ko có thì trả về -1.  -capacity(): độ dài có thể chứa. |
| 34 | **So sánh ArrayList và LinkedList**   |  |  | | --- | --- | | **ArrayList** | **LinkedList** | | Sử dụng 1 mảng động | Sử dụng danh sách liên kết doubly | | Không hiệu quả với thao tác vì cần nhiều chuyển đổi (thêm/xóa) | Hiệu quả trong thao tác (thêm/xóa) | | Tốt hơn trong việc lưu trữ và truy cập dữ liệu | Tốt hơn để thao tác dữ liệu | | Hoạt động như 1 list vì nó implement giao tiếp list | Hoạt động như 1 list và quêu vì nó implement các giao tiếp list và queue | |
| 35 | **Set là gì, các lớp triển khai của Set**  Set là một interface kế thừa Collection interface trong java. Set trong java là một Collection không thể chứa các phần tử trùng lặp.  Set được triển khai bởi **Hashset, LinkedHashset, Treeset hoặc EnumSet.**  **-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 của chúng, về cơ bản là chậm hơn HashSet.  **-LinkedHashSet** được triển khai dưới dạng bảng băm với 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 (thứ tự chèn).  **-EnumSet** là một cài đặt chuyên biệt để sử dụng với các kiểu enum. |
| 36 | **Generic là gì**  Là cơ chế cho phép sử dụng kiểu dữ liệu như là tham số (Tham số hóa dữ liệu) |
| 37 | **Ưu điểm và hạn chế khi dùng generic ?**  1.Phát hiện lỗi tại thời điểm biên dịch.  2.Không cần ép kiểu dữ liệu.  3.Xây dựng các thuật toán tổng quát, tái sử dụng mã nguồn. |
| 38 | **Stack là gì, các phương thức của stack ?**  Stack là 1 ngăn xếp, 1 cấu trúc dữ liệu dạng danh sách thêm và lấy phần tử theo qui tắc FILO(First in last out)  **Các phương thức của stack:**  -size()  -pop(): xem xong và xóa phần tử.  -push(): thêm phần tử vào trên cùng ngăn xếp.  -search()  -peek(): xem phần tử trên cùng của ngăn xếp nhưng không xóa.  -isFull(): kiểm tra ngăn xếp đã đầy hay chưa.  -isEmpty(): kiểm tra ngăn xếp trống hay không |
| 39 | **Queue là gì, các class triển khai của queue ?**  Queue là hàng đợim là 1 cấu trúc dữ liệu dạng danh sách, lấy phần tử theo quy tắc FIFO.  Triển khai của Queue thông qua 3 class:  1.LinkedList  2.ArrayQueue.  3.PriotyQueue. |
| 40 | **Phương thức cơ bản của queue ?**   |  |  | | --- | --- | | List | Queue | | Add() | Offer() thêm phần tử vào | | Element() | Peek() lấy ra xem nhưng ko xóa | | Remove() | Poll() lấy ra xem và xóa | |  |  | |
| 41 | **So sánh Comparable và Comparator, khi nào dùng cái nào ?**   |  |  | | --- | --- | | **Comparable** | **Comparator** | | 1) Bạn phải implements giao tiếp Comparable cho lớp đối tượng cần được so sánh. | Bạn **KHÔNG** phải implements giao tiếp Comparator cho lớp đối tượng cần được so sánh. | | 2) Comparable cung cấp **phương thức compareTo()**để sắp xếp các phần tử. | Comparator cung cấp **phương thức compare()**để sắp xếp các phần tử. | | 3) Comparable thuộc về **java.lang** package. | Comparator thuộc về **java.util** package. | | 4) Chúng ta có thể sắp xếp các 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 Collections.sort(List,Comparator)**. |   **Dùng khi nào?**  Comparable dùng cho trường hợp đơn điều kiện.  Comparator dùng cho nhiều điều kiện. |
| 42 | **Map là gì, các class triển khai**  **Map là gì?**  1.Map được sử dụng để lưu trữ và truy suất theo cặp key-value.  2.Mỗi cặp key-value được gọi là entry.  3. Mỗi key tương ứng với 1 value.  Các cách triển khai của Map là:  **1.HashMap: -** Lưu trữ dữ liệu dưới dạng key-value – chứa key duy nhất – Chỉ có 1 key null, nhiều giá trị null được – Không đảm bảo thứ tự các entry được thêm vào- bản chất vẫn có thứ tự nhưng theo kiểu mã băm hashcode-hashcode trả về vị trí ô nhớ.  **2.LinkedHashMap:** Như hashMap nhưng chỉ có 1 khác nhau lớn nhất là nó duy trì phần tử thep thứ tự chèn.  **3.TreeMap:** Lưu trữ dữ liệu dưới dạng key-value – chứa key duy nhất – Chỉ có 1 key null-Sắp xếp theo thứ tự tăng dần mặc định-đối với kiểu object (người dùng tự định nghĩa) phải implement comparable. |
| 43 | **Cây nhị phân là gì**  Cây tìm kiếm nhị phân là một cấu trúc dữ liệu hiệu quả cho phép chúng ta xây dựng nên một danh sách mà dữ liệu trên đó được sắp xếp:  Nó được gọi là cây nhị phân vì mỗi Node của cây chỉ có tối đa hai con  Giá trị của tất cả các Node ở cây con bên trái phải <= giá trị của Node gốc.  Giá trị của tất cả các Node ở cây con bên phải phải > giá trị của Node gốc.  Tất cả các cây con(bao gồm bên trái và phải) cũng đều phải đảm bảo 2 tính chất trên. |
| 44 | **Trình bày các cách duyệt cây nhị phân.**  Tìm kiếm nhị phân được thực hiện trên mảng đã được sắp xếp bằng cách so sánh phần tử cần tìm với phần tử tại vị trí giữa của tập dữ liệu:   1. So sánh phần tử cần tìm với phần tử giữa mảng. 2. Tìm thấy thì trả về index. 3. Nếu phần tử cần tìm lớn hơn phần tử giữa thì phần tử cần tìm sẽ nằm ở mảng con bên phải của mảng ban đầu. |
| 45 | **Ngoại lệ (Exception là gì) ?**  Là 1 sự kiện bất thường xảy ra trong quá trình thực thi 1 chương trình java, nó làm phá vỡ luồng xử lý bình thường của 1 chương trình, thậm chí chết chương trình. |
| 46 | **Phân loại Exception**  Có 2 loại exception:   1. Checked exception: là loại exception xảy ra trong lúc complier time, loại exception này không thể bỏ qua được trong quá trình complie, bắt buộc phải handle nó. Vdu: FileNotFoundException, DatabaseException,IOException. 2. Unchecked Exception: là loại exception xảy ra tại thời điểm thực thi chương trình, lỗi này không chắc chắn xảy ra. Loại exception này được bỏ qua trong quá trình complie, không bắt buộc phải xử lý. Vdu: ArrayIndexOutOfBandException,NullPointerException. |
| 47 | **Phân biệt Error và Exception**  Một Error là một điều kiện không thể cứu chữa xuất hiện tại runtime, ví dụ: OutOfMemoryError  Exception là các điều kiện mà xuất hiện là do input không phù hợp, hoặc sai… |
| 48 | **Có bao nhiêu cách để xử lý ngoại lệ (handle)**  Có 2 cách để xử lí ngoại lệ:   1. Dùng try-catch có thể có finally để xử lý ngoại lệ ngay tại đoạn mã bị lỗi. 2. Dùng throw hoặc throws để ném ngoại lệ cho logic, phương thức khác xử lí. |
| 49 | **Một số lưu ý khi dùng try-catch**   1. Tại 1 thời điểm chỉ xảy ra một ngoại lệ được ném ra và chỉ có 1 khối catch được thực thi. 2. Thứ tự đặt catch từ chi tiết đến tổng quát, từ con đến cha. 3. Khối lệnh finally thì luôn được thực thi dù có ngoại lệ xảy ra hay không. 4. Một try có thể có 1 hoặc nhiều catch , nếu không có catch thì phải có finally. |
| 50 | **Thuật toán sắp xếp có các loại gì?**  **1.Bubble sort:** So sánh cặp phần tử liền kề nhau và tráo thứ tự nếu chúng không theo thứ tự(tăng, giảm)  **2.Selection sort:** Chọn phần tử nhỏ nhất/lớn nhất trong n phần tử ban đầu, đưa phần tử này về vị trí đầu dãy hiện hành, đồng thời loại bỏ nó ra khỏi danh sách sắp xếp tiếp theo.  Xem dãy hiện hành còn lại chỉ n-1 phần tử của dãy ban đầu, bắt đầu từ vị trí thứ 2 lặp lại quá trình trên cho đến khi dãy hiện hành chỉ còn 1 phần tử.  **3.Insertion sort:** Tạo 1 danh sách con luôn luôn được duy trì dưới dạng đã sắp xếp.  Sắp xếp chèn là chèn thêm 1 phần tử vào 1 danh sách con đã qua sắp xếp ở trên. Phần tử được chèn vào vị trí thích hợp sao cho vẩn đảm bảo rằng danh sách con đó vẩn được sắp xếp theo thứ tự. |
|  | Collection là một interface, trong khi Collections là một lớp. Collecion interface cung cấp các chức năng về cấu trức dữ liệu cho List, Set, Queue. Nhưng lớp Collections là để sắp xếp và đồng bộ các phần tử Collection. |