|  |  |  |
| --- | --- | --- |
| STT | Câu hỏi |  |
| 1 | JAVA là gì ? | * Là ngôn ngữ lập trình hướng đối tượng. |
| 2 | JDK, JRE, JVM | **JDK ( java deve loop ment kit) :**   * Là công cụ hổ trợ lập trình viên phát triển ứng dụng Java.   **JRE (Java virtual enviroment):**  - Môi trường thực thi ứng dụng Java.  **JVM (Java Virtual Machine):**  - Máy ảo Java – thực thi mã Java byte code .Class ra ngôn ngữ máy. |
| 3 | Thông dịch, biên dịch khác nhau như thế nào ? | **Thông dịch:**   * Chạy tới đâu dịch tới đó.   **Biên dịch:**   * Dịch toàn bộ file 1 lần sau đó sử dụng kết quả biên dịch mà không cần dịch lại. |
| 4 | JAVA là ngôn ngữ thông dịch hay biên dịch ? | Java là một ngôn ngữ đa nền tảng và có thể biên dịch hoặc thông dịch tùy thuộc vào cách thực thi của nó. |
| 5 | Java Platform là gì? | **Java platform** :   * Là một môi trường thực thi đa nền tảng cho các ứng dụng Java, được thiết kế hoạt động trên nhiều hệ điều hành và kiến trúc   phần cứng khác nhau. |
| 6 | Các kiểu dữ liệu trong java, và giá trị mặc định khi khai báo. | **Kiểu dữ liệu nguyên thủy:**   | **Kiểu dữ liệu** | **Giá trị mặc định** | **Bộ nhớ** | | --- | --- | --- | | Boolean(true và false) | False | 1 bit | | Char(dùng để lưu trữ kí tự có kích cỡ 2 byte | '\u0000' | 2 byte | | Byte(dùng để lưu kiểu số nguyên có 1 byte(8bit) | 0 | 1 byte | | Short(dùng để lưu kiểu số nguyên có 2byte(16bit) | 0 | 2 byte | | Int(dùng để lưu kiểu số nguyên có 4 byte(32bit) | 0 | 4 byte | | Long(dùng để lưu kiểu số nguyên 8 byte(64bit) | 0L | 8 byte | | Float(dùng để lưu kiểu số thực có 4 byte(32bit) | 0.0f | 4 byte | | Double(dùng để lưu kiểu số thực có 8 byte(64bit) | 0.0d | 8 byte | |
| 7 | Sự khác nhau giữa bộ nhớ heap và stack trong java? | **Heap:**   * Là một khư vực lưu trữ bộ nhớ được chia sẽ giữa các luồng trong ứng dụng 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 |
| 8 | Trình bày các cách khởi tạo một mảng trong JAVA ? | 1. Khởi tạo mảng có kích thước cố định.   Int [] array = new int[5];   1. Khởi tạo mảng có các giá trị ban đầu.   Int [] array = {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 ? | 1. Mảng Java có thể dùng kiểu nguyên thủy và đối tượng. 2. Giá trị mặc định của phần tử trong mảng phụ thuộc vào kiểu dữ liệu của phần tử đó. |
| 10 | OOP là gì ? | **OOP (object oriented progaming):**   * 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 ngôn ngữ lập trình. |
| 11 | Phân biệt class và object | **Class:**   * Là một bản thiết kế để tạo ra đối tượng. * Class định nghĩa các thuộc tính và phương thức của đối tượng.   **Object:**   * Đối tượng là thể hiện cụ thể của một class. |
| 12 | 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. |
| 13 | Cách khai báo constructor và đặc điểm constructor trong JAVA | **Contructor try Java:**   * Tên constructor phải trùng với tên lớp 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 một constructor thì Java sẽ cung cấp 1 constructor mặc định không có tham số và body rỗng. * Trong 1 class thì có thể có nhiều contructor * Có thể gọi constructor khác thông qua từ khóa this và lưu ý this phải được gọi đầu tiên trong nội dung của phươg thức khởi tạo. |
| 14 | Phân biệt constructor và method | |  |  |  | | --- | --- | --- | | 1 | **Constructor** | **Method** | | **Kiểu trả về** | Không có kiểu trả về | Có hoặc không có ( void ) trả về.  Non-access modifier:   * **abstract** * **static** * **final** * **synchronized** * **native** * **strictfp** * **transient** * **volatile** | | **Đặt tên** | Trùng tên với class | Có thể giống(k nên dùng) hoặc khác với tên class | | **Biên dịch** | Nếu không khai báo, mặc định java sẽ cấp 1 constructor rỗng không tham số và body rỗng. | Java không có cung cấp method mặc định. | | **Số lần gọi** | Contructor chỉ được gọi 1 lần khi khởi tạo đối tượng. | Method có thể gọi được nhiều lần. | |
| 15 | "Tính bao đóng là gì ?  Làm sao để thu được tính bao đóng trong java ?" | **Tính bao đóng ( ecapsulacation ):**   * Là kỹ thuật ẩn dấu thông tin của đối tượng, chỉ hiện thị thông tin cần thiết của đối tượng.   **Hình ảnh về tính bao đóng trong Java.**   * Access modifier : prrivite , public , defaul , … * Get, set   Mục đích:   * Cho phép kiểm soát được quyền truy cập vào thuộc tính private. * Kiểm soát giá trị được gán cho thuộc tính. |
| 16 | Tham trị, tham chiếu | **Biến tham trị:**   * Lưu trực giá trị trực tiếp của dữ liệu. * Được truyền vào phương thức dưới dạng giá trị của biến đó không ảnh hưởng đến biến gốc bên ngoài phương thức.   **Biến tham chiếu:**   * Lưu trữ địa chỉ vùng nhớ chứa dữ liệu. * Được truyền vào phương thức dưới dạng địa chỉ của vùng nhớ chứa dữ liêu, và thay đổi dữ liệu trong phương thức cũng   ảnh hưởng đến biến gốc bên ngoài phương thức. |
| 17 | Từ khóa static dùng để làm gì ? | * Static được sử dụng để định nghĩa các biến và phương thức trong một lớp hoặc một hàm. * Biến được khai báo static là các biến tồn tại trong suốt vòng đời của chương trình. * Các phương thức được khai báo bằng static thì có thể được gọi trực tiếp từ lớp mà không cần tạo đối tượng. * Static được dùng để quản lý bộ nhớ của lớp một cách có hệ thống và gọn hơn. |
| 18 | Ràng buộc khi sử dụng static | * Phương thức static chỉ có thể gọi phương thức static khác. * Dùng phương thức thì có thể gọi phương thức biến static * Dung khôi static để khởi tạo giá trị cho biến static |
| 19 | Các loại biến trong JAVA | **Biến local ( biến cục bộ ):**   * Biến khai báo trong method, constructor, block. * Không dùng được access modifier và cần khởi tạo gá trị mặc định. * Biến local được tạo khi method, constructor, block thực hiện 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 constructor, 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. |
| 20 | Trình bày các loại access modifier, và phạm vi truy cập. | **Private** : Trong class  **Default**: Trong class, trong package, ngoài parkage nếu class có mối quan hệ kế thừa.  **Public:** Toàn bộ project có thể truy cập được. |
| 21 | Kế thừa 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 code.   **\*Java không hổ trợ đa kế thừa** |
| 22 | 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 là 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. |
| 23 | 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. |
| 24 | Khái niệm đa hình | * Là khả năng 1 đối tượng có thể hiện hành vi theo nhiều cách khác nhau phụ thuộc và ngữ cảnh. |
| 25 | Phân biệt Overloading và Overriding | |  |  | | --- | --- | | **Overloading** | **Overriding** | | Thể hiện đa hình tại compile time | Thể hiện đa hình tại runtime | | 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 một class | Xảy ra ở 2 class có quan hệ kế thừa | |
| 26 | ép kiểu là gì ? các loại ép kiểu | **Ép kiểu ( type casting )**   * Là quá trình chuyển đổi giá trị của một biến từ kiểu dữ liệu này sang kiểu dư liệu khác.   **Ép kiểu ngầm định (implicit type casting)**  - Đây là loại ép kiểu được thực hiện tự động của java khi các kiểu dữ liệu khác nhau được sử dụng trong một biểu thức  **Ép kiểu tường mình (explicit type casting)**   * Đây là loại ép kiểu được thực hiện bằng cách sử dụng toán tử ép kiểu. |
| 27 | Tính trừu tượng là gì ? | **Tính trừu tượng (Abstraction)**   * Là khả năng ẩn dấu các chi tiết của trình triển khai, chỉ hiển thị tính năng, kết quả cho người dùng. |
| 28 | Phân biệt abstract class và interface | |  |  | | --- | --- | | **Abstract class** | **Interface** | | Là một lớp từu 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 phương thức trừu tượng và phương thức không trừu tượng. | Không thể chứa 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 của interface để có thể sử dụng được. | |
| 29 | So sánh Array và ArrayList | |  |  | | --- | --- | | **Array thường** | **Arraylist** | | Có kích thước cố định | Kích thước có thể thay đổi | | Chỉ có 1 thuộc tính length | Có nhiều phương thức | | Có thể lưu kiểu nguyên thủy và đối tượng | Chỉ lưu được kiểu đối tượng | | Tốc độ lưu trữ nhanh | Tốc độ lưu trữ chậm | |
| 30 | So sánh ArrayList và LinkedList | |  |  | | --- | --- | | **Arraylist** | **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 | |
| 31 | So sánh List, Set và 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 đưuọc sử dụng để thưc hiện các thao tác “đưa vào stack” và “lấy ra từ stack” | | Các phần tử trong List được truy cập thông qua chỉ mục (Index) | Không có chỉ mục để truy cập các phần tử trong Set | Có hai loại Queue:   * Queue thông thường * Dequeue( Double Ended Queue ) | Chỉ có thể truy cập phần tử cuối cùng ( đỉnh của Stack). | | Các lớp cơ sở của List:   * List * Arraylist * LinkedList * Vector | Các lớp cơ sở của Set:   * HashSet * TreeSet * LinkedHashSet | Các lớp cơ sở của Queue:   * PriorityQueue * LinkedList | Các lớp cơ sở của Stack:   * Stack | |
| 32 | 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:** HashSet không đảm bảo thứ tự các phần tử. * **LinkedHashSet:** Có thể truy cập các phần tử theo thứ tự chúng được theo vào. * **TreeSet :** Đảm bảo rằng các phần tử được sắp xếp theo thứ tự tăng dần hoặc giảm dần. |
| 33 | Generic là gì | * Là cơ chế cho phép sử dụng biến dữ liệu như là tham số (Tham số hóa kiểu dữ liệu ). |
| 34 | Ưu điểm và hạn chế khi dùng 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.   **Hạn chế:**   * Có thể làm cho mã nguồn khó hiểu hơn * Có thể gẩy ra lỗi trộn lẫn kiểu dữ liệu. |
| 35 | Stack là gì, các phương thức của stack ? | **Stack:**   * Là một cấu truc dữ liệu danh sách thêm và lấy ra phần tử theo quy tắc vào trước ra sau (FILO).   **Phương thức:**   * Push(E item) : Thêm phần tử vào đầu ngăn xếp. * Pop() : Xóa phần tử trên cùng 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 ngắn xếp. * Empty(): Kiểm tra xem ngăn xếp có trống không. * Search(Object o): 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, phương thức trả về -1. |
| 36 | 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:**   * LinkedList. * PriortyQueue. * Array De Queue. |
| 37 | Phương thức cơ bản của queue ? | **Phương thức:**   * Add( element ) : Thêm phần tử vào cuối hàng đợi, nếu không thành công sẽ ném ra một ngoại lệ. * Offer( element ) : Thêm phần tử vào cub ối 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 sẽ ném ra một ngoại lệ. * Poll() : Lấy phần tử ở đầu hàng đợi và loại bỏ nó, nếu hàng đợi rỗng sẽ ném ra một ngoại lệ. * Element(): Lấy phần tử đầu hàng đợi nhưng không loại bỏ nó, nếu hàng đợi rỗng sẽ ném ra một ngoại lệ. * Peek() : Lấy một phần tử ở đầu hàng đợi nhưn không loại bỏ nó, trả về null nếu hàng đợi rỗng. |
| 38 | 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). | | **Sử dụng**: khi muốn 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 Comparable. | |
| 39 | Map là gì, các class triển khai | **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 2 key trùng lặp nhau, mỗi key tương ứng 1 value.   **Các class triển khai:**   * **HasMap**   **-** Giống đặc điểm Map.  \*Điểm khác:  - Không đảm bảo được các entry được thêm vào.  - cho phép một 1key value là null.  - cho phép nhiều value có thể là null.   * **LinkedHashMap:**   - Giống với hashMap  \*Điểm khác:  - Duy trì các phần tử entry theo thứ tự chèn vào.   * **TreeMap:**   **-** Giống đặc điểm Map.  \*Điểm khác:  - Không có key giá trị null.  - Duy trì các phần tử được thêm vào theo thứu tự key được sắp xếp ( mặc định là sắp xếp tăng ). |
| 40 | Phương thức equals() và hashcode() trong Java là gì? | Equals() : Là phương thức để so sánh hai đối tượng có bằng nhau hay không.  HashCode() : là Phương thức để tính toán mã băm của một đối tượng. |
| 41 | Cây nhị phân là gì | **Cây nhị phân ( binary tree ):**   * Là một cấu truc dữ liệu cây trong đó mỗi nút của cây có tối đa hai nút con,được gọi là nút trái và nút phải. * Một cây nhị phân có thể rỗng hoặc chỉ có một nút gốc ( root node ). |
| 42 | Trình bày các cách 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. |
| 43 | Ngoại lệ (Exception là gì) ? | * Là sự kiện bất thường, xẩy ra trong quá trình thực thi một chương trình Java, nó phá vỡ luồng xử lý bình thường của   chương trình thậm chí chết trương trình. |
| 44 | Phân loại Exception | **Checked Exception:**   * Loại exception xấy ra cao, bắt lỗi tại compileTime * Không thể bỏ qua trong lúc Compile. * Bắt buộc lặp trình viên phải handle.   **\*Ví dụ:**  1**.** IOException **:** Được ném ra khi có lỗi trong quá trình xử lý đọc/ghi tệp tin hoặc xử lý mạng.  2. ClassNotFoundException : Được ném ra khi class không thể đưuọc tìm thấy.  3. SQLException **:** Được ném ra khi có lỗi trong quá trình xử lý cở sở dữ liệu.  4. FileNotFoundException : Được ném ra khi một chương trình cố gắng mở một file để đọc dữ liệu nhưng không tìm thấy.  **UnChecked Exception:**   * Loại exception xấy ra tại thời điểm thực thi của chương trình ( Runtime ) * Là Loại lỗi không chắc chắn xẩy ra có thể bỏ qua trong quá trình compile. * Lập trình viên không bắt buộc phải handle.   **\*Ví dụ:**  1. NullPointException: Ném ra khi một tham chiếu đến đối tượng null.  2. ArrayIndexOutOfBoundsException: Ném ra khi một chỉ mục mảng không hợp lệ được sử dụng.  3. ClassCastException: Ném ra khi một đối tượng không thể ép kiểu sang kiểu khác.  4. InputMissmatchException: Ném ra khi người dùng nhập ko hpải số nguyên.  5. NoSuchElementException: Ném ra khi người dùng ko nhập bất kỳ gì và nhấn Enter.  6. NumberFormatException: Ném ra khi người dùng nhập sai định dạng của kiểu dữ liệu. |
| 45 | Phân biệt Error và Exception | **Error:**   * Là những lỗi nghiêm trọng không thể khắc phục trong quá trình chạy chương trình.   **\*Ví dụ:**  1. OutOfMemoryError : Khi hết bộ nhớ.  2. StackOverflowError : Khi bị tràn ngăn xếp.  **Exception:**   * Là những lỗi có thể xử lý được trong quá trình chạy chương trình. |
| 46 | Có bao nhiêu cách để xử lý ngoại lệ (handle) | **1 .Sử dụng cấu trúc try – catch:**   * Đặt đoạn mã có khả năng xẩy ra ngoại lệ trong khối try, sau đó bắt ngoại lệ bằng cách sử dụng khối catch tương ứng với loại ngoại lệ.   **2 .Ném ngoại lệ ( throw):**   * Ta có thể ném ngoại lệ bằng từ khóa throw trong trường hợp không muốn hoặc không thể xử lý ngoại lệ tại chổ. |
| 47 | Một số lưu ý khi dùng try-catch | * Khối try có thể không có khối catch. * Tại một thời điểm thì chỉ xẩy ra một ngoại lệ, và chỉ có 1 khối catch được thực thi. * Khối ctach phải đưuọc sắp xếp từ cụ thể đến bao bọc lớn nhất. |
| 48 | Serialization là gì ? lưu ý khí sử dụng Serialization | **Serialization:**   * Là cơ chế tuần từ hóa cho phép chuyển trạng thái của 1 dối tượng vào trong byte stream.   **Deserialization:**   * Là chuyển đổi từ byte stream về object.   **\*Lưu ý khi sử dụng Serialization.**  **-** Cần triển khai implêmnt Interface Serialization cho lớp đối tượng cần triển khai đọc ghi byte stream.  - Thuộc tính của đối tượng phải Implêmnt Serialization.  - Thuộc tính áp dụng từ khóa static transient thì sẽ không được Serialization.  - Class cha Implement Serializaton thì lớp con không cần Implêmnt nữa. |

**BỘ CÂU HỎI 48 CÂU CƠ BẢN**

**PHỎNG VẤN MODULE 2**

**ADVANCED PROGRAMMING WITH JAVA**