**BỘ CÂU HỎI PHỎNG VẤN MODULE 2**

**ADVANCED PROGRAMMING WITH PHP**

|  |  |
| --- | --- |
| **STT** | **Câu hỏi** |
| **Thành thạo cú pháp ngôn ngữ PHP** | |
| 1 | **PHP có mấy cách khai báo?**  + Cách thường dùng: <?php ?> + <? ?> Muốn sử dụng cách này phải cài đặt short\_open\_tag = On trong php.ini + <?= ?> : Để echo biến + <script language=”php”> </script> Cách này được xoá từ phiển bản php 7.0.0 trở đi |
| 2 | **Hằng trong PHP khác gì so với biến? Nếu 1 hằng được định nghĩa 2 lần, thì liệu có bị lỗi không?**   |  |  | | --- | --- | | **Hằng** | **Biến** | | Hằng cũng lưu trữ thông tin nhưng giá trị không thay đổi được  Khi đã định nghĩa hằng thì ko được định nghĩa lại, nếu ko sẽ bị lỗi.  Cú pháp: const tenhang hoặc sử dụng hàm define(“tenhang”, “giatri”);  Gọi: echo tenhang hoặc echo constant(“tenhang”)  Phạm vi: Hằng số là toàn cục | Biến dùng để lưu trữ thông tin và có thể thay đổi được.  Ko cần đượ định nghĩa, có thể thay đổi giá trị.  Cú pháp $tenbien  Gọi: echo $tenbien;  Phạm vi:  +Toàn cục: Khai báo ngoài hàm, chỉ truy cập ngoài hàm  +Địa phương: Khai báo trong hàm, chỉ truy cập trong hàm đó.  + Static: Dùng để lưu lại giá trị biến bị xoá sau khi thực thi trong trong hàm xong | |
| 3 | **Phân biệt $\_POST và $\_GET trong php?**   |  |  | | --- | --- | | **POST** | **GET** | | Dữ liệu đươc gửi đi KHÔNG hiển thị trên thanh địa chỉ của trình duyệt  Được sử dụng để gửi các dữ liệu nhạy cảm: mật khẩu, tài khoản …  Không hạn chế dung lượng dữ liệu gửi đi. Thường được sử dụng để gửi dữ liệu lớn lên server. | Dữ liệu được gửi đi sẽ hiển thị trên thanh địa chỉ của trình duyệt  Không nên sử dụng để gửi các dữ liệu nhạy cảm (chẳng hạn như mật khẩu)  Thường được sử dụng để gửi những dữ liệu nhỏ (Tối đa 2048 ký tự) |   Giống nhau: GET và POST đều là hai phương thức của giao thức HTTP.  Đều gửi dữ liệu về server xử lí, sau khi người dùng nhập thông tin vào form và thực hiện submit. |
|  |  |
| 4 | **Phân biệt continue, return, break**   |  |  |  | | --- | --- | --- | | **Continue** | **Return** | **Break** | | Được sử dụng trong vòng lặp, và thoát khỏi vòng lặp hiện tại , kiểm tra điều kiện lần lặp kế tiếp. Khối lệnh phía sau continue sẽ ko được thực thi. | Sẽ trả về một giá trị và thoát đoạn mã hiện tại | Dùng để thoát khỏi vòng lặp (for, foreach, while, do-while) cho dù vòng lặp chưa kết thúc. Ngoài ra còn thoát khỏi switch-case | |
| 5 | **Mảng là gì? Có mấy loại mảng trong PHP?**  Mảng là một biến đặc biệt có thể chứa được nhiều phần tử, từ đó ta có thể dễ dàng lưu trữ, xắp xếp, hay xóa bỏ các phần tử trong mảng một cách dễ dàng.   |  |  |  | | --- | --- | --- | | **Mảng chỉ số là số (index)** | **Mảng liên hợp (associative)** | **Mảng nhiều chiều (Multidimensional)** | | Là mảng mà mỗi phần tử của nó có chỉ số là 1 số nguyên  Khai báo: $name = [“Tam”, “Le”, “Duc” ]; | Là mảng mà chỉ số có dạng chuỗi kí tự  $ages = array("Peter"=>22, "Clark"=>32, "John"=>28); | Mảng đa chiều là một mảng chứa một hoặc nhiều mảng.  $arrr = [“Tam” => [“Duc”, “Le”]]; | |
|  |  |
| 6 | **Nêu các cách duyệt mảng PHP**  Sử dụng vòng lặp for, foreach, truy cập đến từng phần tử qua chỉ số để duyệt mảng.  <?php foreach ($array as $key => $value) {  $array[$key]  } Foreach ($arr as $value) {  $value } $arr[0], $arrr[*‘Tam’*]; |
| 7 | **Mảng 2 chiều là gì? Lấy ví dụ?**  Mảng 2 chiều là mảng trong nó giá trị của chỉ số là 1 mảng 1 chiều.  VD:  <?php [[*“Code”*, *“Gym”*], [*“Ha”*, *“Noi”*].  $cars = array (  array("Volvo",22,18),  array("BMW",15,13),  array("Saab",5,2),  array("Land Rover",17,15)  ); // 2 chieu [1, [1], [1=>[1]]]; // 3 chieu //$a[0] = 1;  //$a[1][0] = 1;  //$a[2][1][0] = 1; |
| 8 | **Sự khác nhau giữa mảng liên kết và mảng một chiều**   |  |  | | --- | --- | | **Mảng liên kết** | **Mảng 1 chiều** | | Duyệt mạng dùng for, foreach  Các chỉ số của mảng là chuỗi ký tự | Duyệt mạng dùng for, foreach  Chỉ số của mảng là dạng số | |
| 9 | **Hàm là gì? Nếu cú pháp khai báo hàm**  là tập hợp các đoạn mã thực hiện một nhiệm vụ cụ thể có thể tái sử dụng.  Cú pháp khai báo:  function tenham(){  //đoạn mã thực thi  } |
| 10 | **Trong PHP để gộp mảng ta dùng hàm gì? Để tách mảng ta dùng hàm gì?**  **Gộp mảng dùng hàm**  array\_merge  https://www.php.net/manual/en/function.array-merge.php  array\_combine ( array $keys , array $values ) <https://www.php.net/manual/en/function.array-combine.php>  **Tách mảng**:  Array\_chunk : Tách ra các mảng con theo kích cỡ phẩn tử.  array\_slice(array,start,[length],[preserve]): $input = array("a", "b", "c", "d", "e"); print\_r(array\_slice($input, 2, -1));  Array  (  [0] => c  [1] => d  ) |
| 11 | **Theo bạn, sự khác nhau của toán tử & (AND) và && trong PHP là gì?**  Toán từ & và && trong PHP đều là **phép toán AND**, tuy nhiên toán tử một dấu & áp dụng **theo kiểu bit**, nói dễ hiểu hơn một dấu & là phép AND thao tác trên các bit ví dụ 1 & 0 thì ra 0. Phép toán hai dấu & thì chỉ áp dụng cho kiểu boolean True và False.  <?php $a = 100; $b = 50;  // Check two condition using  // AND operator  if ($a == 100 and $b == 10)  echo "True"; else  echo "False";  🡺 Output: False  <?php $bool = TRUE && FALSE;  // Display the result of && operation  echo ($bool ? 'TRUE' : 'FALSE'), "\n";  // Expression to use AND operator  $bool = TRUE and FALSE;  // Display the result of AND operation  echo ($bool ? 'TRUE' : 'FALSE');  🡺Output : FALSE  TRUE |
| 12 | **Hãy cho biết $a++ và ++$a khác nhau ở đâu?**  Cả hai đều được dùng để tăng thêm một đơn vị cho biết số  Tuy nhiên điểm khác biệt ở $a++ được thực tăng sau khi nó được gọi, còn ++$a được thực thi ngay khi nó được gọi.  Ví dụ $a bằng 5, khi dùng hàm echo để in nó ra với giá trị a++ thì nó sẽ ra là 5, còn khi dùng echo cho ++$a thì giá trị của được in ra sẽ là 6. |
| 13 | **Sự khác nhau giữa == và ===**  == là so sánh về giá trị, còn === là so sánh về cả giá trị lẫn kiểu dữ liệu  VD: 1 == “1” là TRUE, nhưng 1=== “1” là FALSE. |
| 14 | **Sự khác nhau giữ emty() và isset()**  Isset(): Xác định xem 1 biến đã được khai báo chưa (Sẽ trả về TRUE, nếu biến không là NULL)  Empty(): Kiểm tra xem biến có phải là rỗng hay ko. |
| 15 | **$$ có nghĩa là gì**  Biến của biến: (variable variables)  VD:  <?php $real\_variable = 'test'; $name = 'real\_variable'; echo $$name;  OUTPUT: test |
| 16 | **Tham số là gì?**  là những biến được khai báo trong hàm. |
|  |  |
| **Sử dụng được try-catch, xử lý được ngoại lệ** | |
| 17 | **Ngoại lệ là gì? Tại sao phải xử lý ngoại lệ?**  **Ngoại lệ** là các lỗi phát sinh trong quá trình thực thi.  VD: Lỗi khi máy tính xử lý chia cho 0.  **Tại sao phải xử lý ngoại lệ:** Để ta xử lý những lỗi bất ngờ xảy ra trong khi chạy chương trình |
| 18 | **Sử dụng gì để bắt ngoại lệ?**  Cú pháp bắt ngoại lệ:  <?php Try {  //Câu lệnh có thể xảy ra ngoại lệ. { Catch (*Exception* $e) {  // Câu lệnh hiển thị  } |
| 19 | **Liệt kê các loại lỗi trong PHP.** + Lỗi cú pháp: xảy ra tại thời điểm biên dịch chương trình.  + Lỗi khi chạy chương trình: Lỗi khi chạy chương trình trong suốt quá trình thực thi.  + Lỗi logic:  Lỗi về tính cấu trúc của chương trình, khó có thể tìm ra dấu vêt.  Xảy ra khi bạn tạo một lỗi về dnh logic mà điều khiển script của bạn và không nhận được kết quả như mong đợi.  Khó nắm bắt được các lỗi này, bởi vì nó phụ thuộc vào yêu cầu và kiểu logic mà bạn đặt vào chương trình. |
|  |  |
| **Trình bày được mô hình Lập trình Hướng đối tượng** | |
| 20 | **Nêu 4 đặc trưng của OOP? Lấy ví dụ**  **Tính đóng gói:** Hạn chế truy cập vào thuộc tính, phương thức của đối tượng (lớp) thông qua việc giới hạn quyền truy cập (hoặc thay đổi).  VD: Ta thấy một viên thuốc chữa cảm. Chúng ta chỉ biết nó chữa cảm sổ mũi nhức đầu và một số thành phần chính, còn cụ thể bên trong nó có những hoạt chất gì thì hoàn toàn không biết.  VD2: Khi rút tiền ATM, ta chỉ quan tâm tới số tiền ban đầu, số dư sau khi rút mà ko quan tâm tới quá trình bên trong ATM xảy ra như thế nào.  **Tính kế thừa**: Cho phép 1 lớp có thể kế thừa thuộc tính, phương thức từ lớp khác đã định nghĩa.  VD: 2 lớp Android, iPhone  Mỗi lớp đều đại diện cho một loại smartphone khác nhau nhưng lại có những thuộc tính giống nhau như gọi điện, nhắn tin, chụp hình. Thay vì sao chép những thuộc tính này, ta nên đặt chúng vào một lớp chung gọi là lớp cha. Chúng ta có thể định nghĩa lớp cha – trong trường hợp này là Smartphone và có những lớp con kế thừa từ nó, tạo ra một mối quan hệ cha/con.  **Tính trừu tượng**: Giúp giảm sự phức tạp thông qua tập trung vào đặc điểm trọng yếu hơn là đi sâu vào chi tiết.  Trừu trượng hoá các thực thể ở đời sống vào trong lập trình thông qua các đoạn mã.  VD: Con chó có các hành động : Tính kêu, chạy, ăn thức ăn gì nhưng ko quan tâm đến thịt nó có ngon ko. VD2: Bài toán quản lý sinh viên chúng ta chỉ cần quản lý các thông tin như Họ tên Ngày sinh Giới tính … Điểm thi mà lại không cần quản lý thêm các thông tin: Màu tóc Sở thích Chiều cao  **Tính đa hình**: Các đối tượng có cùng thuộc tính, phương thức nhưng khi thực hiện các đối tượng cho ra kết quả khác nhau.  VD: Ta có 2 con vật: chó, mèo. Cả 2 con vật này đều là lớp động vật. Nhưng khi ta bảo cả 2 động vật kêu thì con chó sẽ kêu gâu gâu, con mèo sẽ kêu meo meo. |
| 21 | **Trình bày khái niệm OPP? Tại sao lại sử dụng OOP** |
|  | **Lập trình hướng đối tượng** ra đời giải quyết các vấn đề mà lập trình truyền thống gặp phải. Lập trình hướng đối tượng không chỉ đơn giản là các cú pháp, câu lệnh mới mà còn là một cách tư duy mới khi giải quyết một vấn đề.  Bao gồm 4 tính chất TÍnh đóng gói, tính kế thừa, tính đa hình, tình trừu tượng  **Tại sao lại sử dụng OOP:**  Tại sao phải sử dụng oop   * Bảo trì nhanh chóng, hiệu quả * Dễ Chia hệ thống các phần nhỏ để xử lý. * Tích hợp tốt với các hệ thống máy tính đã có sẵn. * Tích hợp tốt với các hệ điều hành hiện đại. * Khả năng tạo giao diện người dùng trực quan. |
| **Sử dụng được lớp, đối tượng, phương thức, thuộc tính, hàm tạo** | |
| 22 | **Hàm contruct được dùng để làm gì**  Hàm construct là hàm khởi tạo của lớp, được sử dụng để gọi các câu lệnh bên trong nó khi khởi tạo 1 đối tượng. |
| 23 | **Liệt kê access modifer?**  Public, protected, private  VD: |
| 24 | **Phân biệt static với seft**  static được dùng cho các thuộc tính, phương thức. Truy cập được thuộc tính phương thức trong lớp đó, lớp kế thừa sẽ bị lỗi.  seft để gọi các thuộc tính, phương thức trong 1 lớp. Được sử dụng trong cả lớp kế thừa và lớp hiện tại.   * **Self**: Truy xuất đến class khai báo nó. * **Static**: Truy xuất đến đối tượng hiện tại. * (Phương thức static thì ko thể sử dụng từ khoá $this trong đó bởi vì nó chỉ gọi bằng từ khoá self đại diện truy xuất đến class). |
| 25 | **Đâu là cách quản lý package trong PHP.**  Một **namespace** trong PHP được sử dụng để đóng gói các mục tương tự như trừu tượng hóa trong các khái niệm lập trình hướng đối tượng.  Namespace được sử dụng để thiết lập mối quan hệ giữa các lớp, hàm và hằng.  Một namespace có thể được xác định bằng cách sử dụng từ keyword namespace được dành riêng cho PHP để sử dụng nội bộ để tránh xung đột với các định danh do người dùng tạo ra. |
| 26 | **Lớp là gì? Cách khai báo lớp**  **Lớp** là một bản thiết kế để tạo các đối tượng (một cấu trúc dữ liệu cụ thể), cung cấp các giá trị ban đầu cho trạng thái (các biến hoặc thuộc tính) và thực hiện hành vi (hàm hay phương thức).  **Cách khai báo lớp**:  <?php  Class tenlop {  //Khai báo thuộc tính, phương thức } |
| 27 | **Triển khai được cơ chế nạp chồng phương thức (overloading)**  Overloading cho phép một lớp có khả năng định nghĩa ra nhiều phương thức có cùng tên, nhưng khác nhau về tham số truyền vào. |
| 28 | Class và object giống và khác nhau thế nào?   |  |  | | --- | --- | | **Class** | **Object** | | Class là bản thiết kế chi tiết từ đó để tạo các đối tượng  Một lớp được đưa ra các phương thức để liên kết với nhau .  Các lớp có sự tồn tại logic  Ko chiếm bộ nhớ khi 1 lớp  Lớp khai báo 1 lần | Đối tượng là thể hiện của lớp, giúp người lập trình sử dụng các thuộc tính và phương thức từ bên trong lớp.  Đối tượng hoạt động như 1 biến của lớp.  Các đối tượng có tồn tại vật chất.  Chiếm bộ nhớ khi khởi tạo 1 đối tượng.  Đối tượng có thể khởi tạo nhiều lần. | |
| 29 | **Constructor là gì**?  Là hàm khởi tạo trong lớp, khi khởi tạo đối tượng thì hàm này luôn được chạy. |
| 30 | **Các thành phần của một lớp?**  Tên lớp, thuộc tính, đối số, phương thức, các mức truy cập, kiểu dữ liệu của thuộc tính. |
| 31 | **Vẽ sơ đồ lớp của lớp sau: tên lớp Customer: name, birthday; method: getInfo() : đưa ra tên + tuổi**   |  | | --- | | Customers: | | $name string  $birthday string | | getInfo() | |
| 32 | **Làm thế nào để khởi tạo đối tượng**?  Giả sử có 1 lớp Customers  Khởi tạo: $customer = new Customers(); |
| 33 | **Có mấy cách khai báo lớp**?  <?php  class MyClass {  // field, constructor, and   // method declarations }  Dùng từ khoá Extends  class ExtendClass extends SimpleClass |
| 34 | **Khai báo lớp Customer vừa nêu**  <?php  class Customer {  protected $name;  protected $birthday;  public function \_\_construct ($name, $birthday){  $this->name = $name;  $this->birthday = $birthday;  }   Public function getInfo() {  Return "Name=".$this->name.",Birthday=".$this->birthday;  } } |
| 35 | **Nêu vòng đời của đối tượng**  1. Khởi tạo đối tượng.  2. Sử dụng đối tượng (sử dụng các thuộc tính của đối tượng, gọi các phương thức của đối tượng).  3. Huỷ đối tượng (diễn ra khi đối tượng không còn được sử dụng). |
| 36 | **Làm thế nào để truy xuất thuộc tính của đối tượng**?  Sử dụng $this->thuoctinh (trong lớp), self::$thuoctinh (trong lớp), tenlop::$thuoctinh (ngoài lớp)   |  |  | | --- | --- | | Thuộc tính trong lớp | Thuộc tính ngoài lớp | | * Truy xuất thuộc tính trong lớp bằng cách sử dụng $this->thuoctinh | * Khởi tạo đối tượng, nếu thuộc tính khai bao trong lớp là public thì ta có thể truy xuất đến được | |
|  |  |
| **Triển khai được cơ chế kế thừa** | |
| 37 | **PHP có đa kế thừa không?**  Đa kế thừa có nghĩa là một lớp có thể kế thừa nhiều hơn một lớp khác, điều này là ko thể.  Trong trường hợp nếu kế từ Interface thì ta có thể implements nhiều hơn 1 interface. Ngoài ra có thể sử dụng Trait  <?php  interface Health {  public function showWeight($weight); }  interface Action {  public function speak(); }  class Cat implements Health, Action {  public $weight;  private $say;   public function \_\_construct()  {  $this->weight = 20;  $this->say = 'meow';  }   public function showWeight($weight)  {  echo $weight;  }   public function speak()  {  echo $this->say;  } }  $cat = new Cat();  $cat->showWeight($cat->weight); // 20 $cat->speak(); // meow  Vì vậy PHP ko có đa kế thừa. |
| 38 | **Thế nào là ghi đè?**  **Tạo ra một phương thức trong lớp con có trùng tên, cùng tham số, cùng kiểu trả về với một phương thức đã được tạo ở lớp cha.**  VD:  <?php  class Foo {  function myFoo() {  return "Foo";  } }  class Bar extends Foo {  function myFoo() {  return "Bar";  } }  $foo = new Foo; $bar = new Bar; echo($foo->myFoo()); //"Foo" echo($bar->myFoo()); //"Bar" ?> |
| 39 | **Phân biệt abtrast class và interface, khi nào sử dụng chúng?**   |  |  | | --- | --- | | **Interface** | **Abstract** | | Interface là "bản thiết kế" cho **Method**:  Interface thường được sử dụng trong trường hợp các class **kế** **thừa không có cùng bản chất** (nhóm đối tượng) nhưng chúng có thể thực hiện các **hành động giống nhau**. (đa hình)  Interface không phải là **1 lớp.** Nó được mô tả như là 1 bản thiết kế cho các class có chung cách thức hoạt động.  Vì không phải là 1 lớp nên **không thể định nghĩa các thuộc tính, khởi tạo đối tượng** mà chỉ khai báo các phương thức.  Các phương thức chỉ khai báo tên hàm và **không viết nội dung hàm trong đó**. (ko có phần thân)  Không có khái niệm phạm vi của phương thức, tất cả đều là **public.**  Lớp con kế thừa từ interface sẻ phải **override** tất cả các phương thức trong đó.  Một lớp có thể kế thừa từ nhiều interface khác nhau bằng từ khóa **implements.**  Interface có khai báo hằng số thì lớp con ko thể override được hằng số. | Abstract Class là "bản thiết kế" cho **Class**:  Abstract thường được sử dụng trong trường hợp các class kế thừa từ nó có **cùng bản chất** (thuộc 1 nhóm đối tượng)  Lớp trừu tượng được xem như **một class** cha cho tất cả các Class có **cùng bản chất**.  Lớp trừu tượng có thể khai báo **các thuộc tính và các phương thức bình thường** và không có thuộc tính trừu tượng (lớp trừu tượng không cho phép khởi tạo tham số, chỉ khai báo mà thôi).  Bên cạnh đó nó không cho phép tạo instance, nghĩa là sẽ **không thể tạo được các đối tượng** thuộc lớp đó.  Lớp con kế thừa có **phạm vi phải hơn hoặc bằng lớp cha**.  Lớp con kế thừa từ lớp trừu tượng có thể **override** các phương thức trừu tượng trong đó.  Các phương thức được định nghĩa là abstract trong abstract class thì chỉ khai báo tên hàm và không viết nội dung hàm trong đó. Nội dung sẻ được viết trong class kế thừa khi **override** lại phương thức đó.  Do đó mỗi lớp dẫn xuất (lớp con) chỉ có thể kế thừa từ một lớp trừu tượng bằng từ khóa **extends**. | |
|  |  |
| **Sử dụng được UML (\*)** | |
| 40 | **UML là gì**?  là một ngôn ngữ mô hình gồm các ký hiệu đồ họa mà các phương pháp hướng đối tượng sử dụng để thiết kế các hệ thống thông tin một cách nhanh chóng.    https://allaravel.com/blog/lap-trinh-huong-doi-tuong-trong-php-phan-3-da-dang-hoa-trong-ke-thua |
| 41 | **Liệt kê các mối quan hệ giữa các lớp? Giải thích chúng?**  **Các quan hệ:**  Single: Một lớp kế thừa từ chỉ một lớp cha • Multilevel: Một lớp kế thừa từ một lớp cha, lớp cha lại kế thừa từ lớp khác ở trên nó • Hierarchical: Một lớp cha có nhiều lớp con với nhiều level khác nhau • Multiple: Một lớp con kế thừa từ nhiều lớp cha( không dùng trong php) |
|  |  |
| **Tuân thủ Clean Code** | |
| 42 | **Trình bày được tầm quan trọng của Clean Code**  + Dễ đọc:  + Dễ hiểu:  + Dễ bảo trì:  + Dễ mở rộng:  Trình bày tên các lớp, phương thức, thuộc tinh có ý nghĩa để các LTV khác có thể hiểu, code trong 1 phương thường ko quá dài, nếu dài nên cắt bớt tối ưu lại các đoạn mã  Các tên lớp, thuộc tính, phương thức ngắn gọn dễ hiểu, đặt theo các quy tăt chuẩn như quy tắc CamelCase, **Coding Convention**  Trình bày code sạch, khoa học để sau này có thể dễ bảo trì, mở rộng. |
| 43 | **Kể tên một số Coding Convention trong PHP** |
|  | Cú pháp lạc đà (camelCase)  * **Cú pháp**: **Ký tự đầu tiên** của **từ đầu tiên** **viết thường**, **những ký tự đầu tiên** của **những từ tiếp theo** **viết hoa**. * Ví dụ: productName, productPrice, thisIsTheNameFollowTheCamelCase  Cú pháp Pascal (PascalCase)  * **Cú pháp**: **viết hoa chữ cái đầu tiên của mỗi từ.** * Ví dụ: ProductName, ProductPrice, ThisIsTheNameFollowThePascalCase  Cú pháp con rắn (snake\_case)  * **Cú pháp**: **Tất cả các chữ cái** đều **viết thường**, và **các từ** **cách nhau** bởi **dấu gạch dưới**. * Ví dụ: product\_name, product\_price, this\_is\_the\_name\_follow\_the\_snake\_case   – **Tên lớp**: đặt theo **PascalCase**.  – **Tên biến, tên hàm**: đặt theo **camelCase** hoặc **snake\_case**.  – **Hằng số**: đặt theo **UPPER\_CASE**. VD: CLICK\_COUNTER.  – **Tên biến, tên lớp**: thường là **danh từ**, **cụm danh từ** hoặc **tính từ**. VD: UserModel, userName, downloadCounter, isDownloaded.  – **Tên hàm** thường\*\* bắt đầu bằng động từ\*\*. VD: getUserName, setUserName, increaseDownloadCounter.  – **Tên phải có nghĩa, không được đặt tên kiểu viết tắt**. VD: uName, pName, idl, a, a1, doFA.  – **Tránh đặt những tên quá chung chung, tối nghĩa**. VD: top, doIncrease, getAll. |
| **Sử dụng được các kỹ thuật refactoring** | |
| 44 | **Refactor là gì? Em đã sử dụng các kỹ thuật refactor nào**? |
|  | Tái cấu trúc mã nguồn là các kỹ thuật cho phép chỉnh sửa mã nguồn nội bộ mà không làm thay đổi hành vi của hệ thống đối với bên ngoài  Các kỹ thuật refactor:  + Tách biến: Trong nhiều biểu thức có các biến gây khó hiểu  + Tách hằng: Tách những magic value , từ các biểu thức khó khăn khi đọc  + Tách phương thức: Các phương thức quá dài sẽ tạo khó khăn cho người đọc hiểu |
| **Sử dụng được các cấu trúc dữ liệu danh sách** | |
| 45 | **Trình bày khái niệm về data structures**   * Cấu trúc dữ liệu là hình thức tổ chức dữ liệu và cung cấp các phương thức để thao tác với các phần tử * Các khái niệm: • Container: Lớp chứa dữ liệu • Elements: Các phần tử dữ liệu   VD: ArrayList, LinkedList   |  |  | | --- | --- | | **ArrayList** | **LinkedList** | | Truy xuất ngẫu nhiên nhanh | Truy xuất ngẫu nhiên chậm | | Thêm, xoá chậm | Thêm, xoá nhanh | |
| 46 | **Nêu cơ chế của cấu trúc Stack, Queue**   * Stack lưu trữ dữ liệu hoạt động theo nguyễn lý Last-In-First-Out * Queue hoạt động theo nguyên tắc First-In-First-Out |
|  |  |
| **Thiết kế được CSDL Quan hệ** | |
| 47 | **MySQL là gì?**  ***MySQL là một hệ thống quản trị cơ sở dữ liệu mã nguồn mở (gọi tắt là RDBMS) hoạt động theo mô hình client-server***.  Hệ quản trị CSDL: Một ứng dụng máy tính được sd để quản lý CSDL |
| 48 | **‘table’ và ‘field’ là gì?**  Table là bảng của 1 cơ sở dữ liệu.  Field là 1 trường, thuộc tinhs trong 1 bảng |
| 49 | **‘join’ là gì?**  Join cho phép nối nhiều bảng với nhau, có thể có trường chung hoặc ko. |
| 50 | **‘constraint’ là gì?**   Các ràng buộc được sử dụng để xác định những quy tắc cho dữ liệu bên trong một cái bảng, nó sẽ giới hạn loại dữ liệu có thể thêm vào bên trong bảng. |
| 51 | **Khoá chính dùng để làm gì? Có mấy khoá chính trong một bảng**   * Khóa chính (Primary Key, hay ràng buộc khóa chính) được sử dụng để định danh duy nhất mỗi bản ghi (dòng) trong bảng của cơ sở dữ liệu. * Ngoài ra, nó còn dùng để thiết lập quan hệ 1-n (hay ràng buộc tham chiếu) giữa hai bảng trong cơ sở dữ liệu. * Dữ liệu (value) của field khóa chính phải có tính duy nhất, và không chứa các giá trị Null. * Mỗi bảng chỉ có một khóa chính * Khóa chính có thể tạo ra dựa trên một cột hoặc nhiều cột (được gọi là khoá chính tổ hợp) của table.   Có khoá chính kết hợp 1 từ 1 trường, và khoá chính tổ hợp kết hợp từ 2 trường tạo 1 thành 1 khoá chinh |
| 52 | **Tính toàn vẹn dữ liệu là gì**?   * Tính toàn vẹn của thực thể : khoá chính ko được phép null * Tính toàn venj veè tham chiếu: Mỗi khoá phụ phair khớp với giá tri khoá chính trong table được tham chiếu. |
| 53 | **Ràng buộc là gì?**  Là tạo các liên kết giữa các khoá chính, ngoại nhằm ràng buộc dữ liệu giữa chúng |
| 54 | **Hãy giải thích ‘index’**   * Index là một cấu trúc dữ liệu được dùng để định vị và truy cập nhanh nhất vào dữ liệu trong các bảng database * Index là một cách tối ưu hiệu suất truy vấn database bằng việc giảm lượng truy cập vào bộ nhớ khi thực hiện truy vấn   **. B-Tree**  **.Hash Index** |
| 55 | **‘auto-increment’ là gì?**  một trường nào đó là tăng tự động thì khi bạn thêm record mới bạn không cần phải truyền data cho nó và nó sẽ tự lấy giá trị lớn nhất tăng lên 1 |
| 56 | **Kể tên các quan hệ giữa các bảng?**  Quan hệ các bảng:  Quan hệ 1-1: Một học sinh có 1 mã ID  Quan hệ 1-n: Một trưởng phòng quản lý nhiều nhân viên, một nhân viên chỉ được quản lys bởi 1 trưởng phòng.  Quan hệ n-n: Một sản phẩm có nhiều đơn hàng. Một đơn hàng có nhiều sản phẩm. |
| 57 | ‘view’ là gì?  View có tác dụng giống như tạo một Table ảo với các Fields và Records mà ta có thể tự định nghĩa và khác hoàn tòa với Table gốc.  View có thể truy vấn lấy ra các dữ liệu từ 1 hay nhiều bảng join với nhau.  Nhưng ko thể chỉnh sửa hoặc sủa từ View. Khi câu trúc bảng gốc thay đổi có thể làm View bị lỗi.’  Khi dữ liệu bảng gốc thay đổi thì dữ liệu trên View cũng thay đổi theo |
|  |  |
| **Sử dụng được SQL** | |
| 58 | SQL là gì?  **SQL** là viết tắt của **Structured Query Language** là ngôn ngữ truy vấn có cấu trúc. |
| 59 | **Muốn lấy dữ liệu dùng câu lệnh nào?**  SELECT \* FROM table |
| 60 | **Viết câu lệnh lấy tất cả học viên trong bảng students có name là Nam**  SELECT \* FROM students WHERE name = "Nam" |
| 61 | **Viết câu lệnh thêm mới học viên: tên, tuổi, địa chỉ vào bảng students**  INSERT INTO students (name, age, address) VALUES ("Tam", "30", "Ha Noi") |
| 62 | **Viết câu lệnh tạo bảng students: name, age, address**  CREATE TABLE Students (  name varchar(255) not null,  age int(3),  address varchar(255) ); |
| 63 | **Nhóm các dữ liệu lại ta dùng cái gì?**  Ta có thể dùng câu lệnh GROUP BY |
| 64 | **Phân biệt các loại join?**  Có 4 loại join phổ biến |
| 65 | **Làm thế nào để lấy 25 phần tử đầu tiên của một bảng**  Tên bảng: Table  Primary key: id  SELECT \* FROM table ORDER BY id ASC LIMIT 25 |
|  | Kể tên các cách để tối ưu hoá câu lệnh sql |
|  |  |
| **Triển khai được kiến trúc phân tầng** | |
| 66 | **Trình bày mô hình MVC** |
|  |  |
| **Trình bày được mô hình Ứng dụng Web (\*)** | |
| 67 |  |
| 68 | **Trình bày các giao thức mạng thông dụng**  HTTP, FTP, SMTP, SSH, DNS … |
| 69 | **Trình bày giao thức HTTP**  + Giao thức truyền tải siêu văn bản  + Dùng để liên hệ thông tin Web server và client.  + Là giao thức Client/Server dùng cho www |
| 70 | **Phân biệt phương thức GET và POST**   |  |  | | --- | --- | | **POST** | **GET** | | Dữ liệu đươc gửi đi KHÔNG hiển thị trên thanh địa chỉ của trình duyệt  Được sử dụng để gửi các dữ liệu nhạy cảm: mật khẩu, tài khoản …  Không hạn chế dung lượng dữ liệu gửi đi. Thường được sử dụng để gửi dữ liệu lớn lên server. | Dữ liệu được gửi đi sẽ hiển thị trên thanh địa chỉ của trình duyệt  Không nên sử dụng để gửi các dữ liệu nhạy cảm (chẳng hạn như mật khẩu)  Thường được sử dụng để gửi những dữ liệu nhỏ (Tối đa 2048 ký tự) | |
| 71 | **Phân biệt giao thức HTTP và HTTPS**  HTTP là tên viết tắt của HyperText Transfer Protocol (giao thức truyền tải siêu văn bản), là một giao thức cơ bản dùng cho World Wide Web (www) để truyền tải dữ liệu dưới dạng văn bản, hình ảnh, video, âm thanh và các tập tin khác từ Web server đến các trình duyệt web và ngược lại.  Còn HTTPS là viết tắt của từ HyperText Transfer Protocol Secure và chính là giao thức HTTP có sử dụng thêm các chứng chỉ SSL (secure Sockets Layer) giúp mã hóa dữ liệu truyền tải nhằm gia bảo mật giữa Web sever đến các trình duyệt web.  Nói cách khác HTTPS là phiên bản HTTP nhưng an toàn hơn, bảo mật hơn. |
| 72 | **Trình bày các thành phần: Browser, Web Server, Request và Response**  **Web Browser** là một ứng dụng phần mềm giúp người dùng truy cập các thông tin có trênWorld Wide Web (WWW). Tại mỗi địa chỉ WWW sẽ là một trang web chứa đựng các hình ảnh, video và văn bản được xác định bằng một URL riêng biệt, giúp cho các trình duyệt web truy xuất và hiển thị chúng tới các thiết bị như điện thoại và máy tính của người sử dụng.  **Web Server** là một phần mềm/phần cứng để cung cấp các tài nguyên web  Web Server tiếp nhận các request và trả về response  Web Server sử dụng giao thức HTTP và một số giao thức liên quan khác  Web Server có thể hoạt động độc lập hoặc có thể kết hợp với các phần mềm khác, chẳng hạn như phần mềm cân bằng tải, phần mềm bảo mật, phần mềm cache...  **Request:**  Khi Client truy cập đến 1 trang web đồng nghĩa gửi 1 request tới web server để tiếp nhận request này.  Các thành phần:  + Method : GET hoặc POST  + URL: địa chỉ định danh  + HTTP version: phiên bản của http đang dùng  + Request header: User agent, Cookie …  **Response**: Khi Client gửi 1 request tới 1 trang web và web server tiếp nhận xử lý trả về 1 response  + HTTP version:  + Status code: 2xx, 3xx, 4xx, 5xx …  + Reason- pharse: Mô tả status-code. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 73 | **Website và web app có gì giống và khác ?**   |  |  | | --- | --- | | **Web app** | **Website** | | Ứng dụng web được thiết kế để tương tác với người dùng.  Người dùng ứng dụng web có thể đọc nội dung của ứng dụng web và cũng có thể thao tác dữ liệu.  Trang web ứng dụng phải được biên dịch trước trước khi triển khai.  Chức năng của ứng dụng web khá phức tạp.  Ứng dụng web có tính tương tác cho người dùng.  VD: facebook, amazon, shopee, tiki, lazada | Trang web về cơ bản chứa nội dung tĩnh.  Người sử dụng website chỉ có thể đọc nội dung website nhưng không được thao tác.  Trang web không cần phải được biên dịch trước.  Chức năng của trang web rất đơn giản.  Trang web không tương tác cho người dùng.  VD: dantri, vnexpress, bongdaso … | |
| 74 | **Khi tắt cookie trên trình duyệt thì có ảnh hưởng tới session không?**  Ko ảnh hưởng bởi vì session nó lưu trữ ở biến $\_SESSION còn cookie lưu ở biến $\_COOKIE.  Và cookie được lưu trữ trên trình duyệt ngừoi dùng, session lưu trữ trên máy chủ. |
| 75 | **Phân biệt Application Server và Web Server**   |  |  | | --- | --- | | **WEB SERVER** | **APPLICATION SERVER** | | Web server chỉ bao gồm web container. | Trong khi application server bao gồm web container cũng như EJB container. | | Web server hữu ích hoặc phù hợp cho nội dung tĩnh. | Trong khi application server được trang bị cho nội dung động. | | Web server tiêu thụ hoặc sử dụng ít tài nguyên hơn. | Trong khi application server sử dụng nhiều tài nguyên hơn. | | Web server sắp xếp môi trường chạy cho các ứng dụng web. | Trong khi application server sắp xếp môi trường chạy ứng dụng của doanh nghiệp. | | Trong web server, đa luồng không được hỗ trợ. | Trong application server, đa luồng được hỗ trợ. | | Công suất của web server thấp hơn app server. | Trong khi công suất của application server cao hơn web server. | | Trong web server, giao thức HTML và HTTP được sử dụng. | Trong app server, GUI cũng như các giao thức HTTP và RPC/RMI được sử dụng. | |
| 76 | **Phân biệt session và cookie?** |

|  |  |
| --- | --- |
| **COOKIE** | **SESSION** |
| Cookie là một đoạn văn bản ghi thông tin được tạo ra và lưu trên trình duyệt của máy người dùng, và truy xuất chúng qua biến $\_COOKIE  Cookie được lưu trữ trên trình duyệt người dùng.  Một cookie có các thông tin như: tên cookie, giá trị của cookie, path, thời gian cookie hết hạn, thời gian cookie được tạo …  Dữ liệu cookie dễ dàng sửa đổi khi chúng được lưu trữ ở phía trình duyệt  Mỗi lần gửi dữ liệu từ form (POST or GET) thì cookies sẽ được trình duyệt gửi đi lên máy chủ.  Tắt trình duyệt đi thì cookie ko tự mất đi | Session là phiên làm việc. Nó là cách đơn giản để lưu trữ 1 biến và khiến biến đó có thể tồn tại từ trang này sang trang khác.  Dữ liệu session được lưu trữ ở phía máy chủ.  Mỗi session chứa thông tin dưới dạng mã hoá, có thể truy xuất qua biến $\_SESSION  Dữ liệu session ko dễ dàng sửa đổi vì chúng  được lưu ở phía máy chủ.  Mỗi lần gửi dữ liệu từ form (POST or GET) thì session sẽ được trình duyệt gửi đi lên máy chủ.  Tắt trình duyệt đi thì session sẽ bị mất. |

**Các kiến thức mà em đã học:**

(1-6**) Tổng quan về web**: Giao thức mạng, giao thức http, mo hình client server, Biến, toán tử, ,

Giao thức http: là giao thưc truyền tải siêu văn bản, hình ảnh, video …

Giao thưc mạng phổ biến là: http, ssh, ftp, smtp, dns …

Mô hình client server: Khi client gửi 1 yêu cầu tới server, web server tiếp nhận và xử lý gửi tới web app , tại đây web app lấy dữ liệu ở database và trả về web server, web server trả về 1 respondse và hiển thị ra cho client.

**Mảng, hàm:** trình bày các loại mảng, 3 loại mảng, mảng có chỉ số là số nguyên, mảng liên hợp mảng có chỉ số là kí tự, mảng đa chiều là mảng mà có ít nhất 1 mảng. Hàm bao gồm các câu lệnh nhằm thực hiện 1 nhiệm vụ nào đó. Cú pháp hàm’; ;…

oop 4 tính chất, access modifier: 4 tính chất oop:

**Tính đóng gói:** hạn chế truy cập vào các lớp thông qua việc giới hạn các mưc truy cập public, protected, private.

VD1: Một viên thuốc chữa cảm cúm, ta quan tâm là viên thuốc này chữa bệnh cảm cúm nhưng ta ko hề biết bên trong chúng có chứa những thứ gì

VD2: Khi đi rút tiền ở cây ATM thì ta quan tâm số tiền ban đầu, và số dư trong tài khoản nhưng ko biết cách ATM hoạt động bên trong như thế nào.

**Tính kế thừa**: cho phép các lớp con có thể kế thừa từ lớp cha thông qua các phương thức thuộc tính từ lớp cha.

VD. Lớp iphone samsung: Đều có các phương thức nhắn tin, gọi điện … chúng ta có thể kế thừa chúng thông qua 1 lớp là Điện thoại.

**Tính đa hình**: Các lớp có cùng thuộc tính, phương thức nhưng khi thực hiện thì chúng cho ra kết quả khác nhau.

VD: Con chó có phương thức kêu gâu gâu, còn con mèo có phương thức kêu meo meo.

**Tính trừu tượng**: cho phép trừu tượng hoá các thực thể trong đời sống vào trong lập trình và chỉ quan tâm tới các đặc điểm trọng yếu.

VD1: Khi mô ta về lớp động vật chó, mèo ta chỉ quan tâm tới màu sắc, cân nặng .. mà ko quan tâm tới thịt chúng ngon hay ko

VD2: Khi ta xét về bài toán sinh viên ta chỉ quan tâm tới các thông tin như tên, tuổi, địa chỉ .. nhhuwng ko quan tâm về màu tóc, cân nặng…

Static: đăj trước thuộc tính, phương thức, ko thẻ dùng $this đẻ gọi tới thuoc tính , phuong thức có static. Muôns gọi được ta sử dụng self (truy xuất tới lớp, còn $this truy xuất tới đối tượng hiện tại). static có thể truy xuất tới tt, pt .

**Có 3 mức truy câp public, protected, private: dùng cho cả thuộc tinh, phương thức**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Public | Protected | Private |
| Trong lớp |  |  | x |
| Kế thừa |  | x | x |
| Ngoaif lớp | x | x | x |

|  |  |
| --- | --- |
| **Interface** | **Abstract class** |
| Là bản thiết cho “method”  Là tập hợp các đối tượng có chung hành động nhưng khi thực hiện cho ra kết quả khác nhau (Đa hình)  Ko thể là final  Mực truy cập bắt buộc là public  Các phương thức có thể có thuoc tinh, nhưng k có phần thân  Các lớp con implements phải có các phuong thuc lớp cha  Một lớp có thể kế thừa nhiều interface  Ko phải 1 lớp do đó chúng ko thể khởi tạo đối tượng | Là bản thiết kế cho class  Là tập hợp các đối tượng cùng bản chất  Ko thể là final  Lớp kế thừa nó phải có mức truy cập lớn hơn của cha  Mỗi absstract class có ít nhất 1 abstract method  Các phương thức có phần thân  Các lớp kês thừa phải có các phương thức abstract  Mỗi lớp kế thừa 1 abstract classs  Là 1 lớp nhưng ko thể khởi tạo đối tượng\ |

ÔOP tạo ra nhằm giải quyết các bài toán trong đời sống, sử dụng 4 tính chât của oop

Lớp là bản thiết kế cho đối tượng.

Đối tượng là thể hiện của 1 lớp có các thuôc jtin phương thức

**Quan hệ giữa các lớp là:**

+ Một lớp được kế thừa từ 1 lớp đã đinh nghĩa

+ Lớp cha có thể có nhiều lớp con khac nhau kế thừa

+ Lớp cha kế thừa 1 lớp khác và lớp cha này có 1 lớp con kế thừa.

**(7-14) Clean code:**

Dễ đọc , dễ hiểu, dễ bảo trì, dễ mở rộng.

**Các cách tái cấu trúc (refactor) :** 3 kĩ thuật

Tách biến: tách những biến gây khó đọc hiểu cho người đọc.

Tách hằng: tách các gia trị magic value

Tách phườn thức: các phương thức nên chỉ thực hiện 1 nv cụ thể

**Try catch:** Là khối câu lệnh dùng để bắt các lỗi xảy ra bất ngờ.

Cú pháp : try {

}

Catch (exception $e) {

Hien thi exxception $e->getMessage()

}

**Cấu trúc dữ liệu Arraylist:** là cấu trúc dữ liệu danh sách kiểu mảng, trong đó chung có các phương thức, thuộc tính ‘

Muốn xoá 1 phần tử vị trí thứ i trong arraylist: ta sẽ dịch chuyển vị tri thứ i + 1 về vi tri i, cứ như vậy, ta sẽ thay thế được vi tri i.

Muốn thêm 1 phần tử vi tri i, ta sẽ dịch chuyển vi tri i sang i+ 1 và chèn vị tri này vào, và dịch chuyển i -1 vêf i

**Cấu trúc dữ liệu. Linkedlist**: là cấu trúc dữ liệu sanh sách bảo gôm node, và dữ liệu, mỗi node gồm dữ liệu của node đó và liên kết tới node iteps theo

|  |  |
| --- | --- |
| Arraylist | Linkedlisst |
| Truy câp các phần tử nhanh | Truy cập phần tử chậm |
| Thêm, xoá chậm vì chúng phải dịch chuyển thay đổi caau trúc phần tử | Thêm xoa nhanh vì chỉ theo dạng tham chiếu |

**Stackqueue tree:**

**Stack** hoạt động theo nguyên tắc, vào trước ra sau, vào sau ra trước

**Queue** hoat động theo nguyên tắc, vào trc ra trc, vào sau ra sau

Trêê là câu trúc dạng cây, có node gốc, và các node cha con …

**Các loại sắp xếp, sắp xếp chọn, chèn, nổi bọt:**

Sắp xếp nổi bọt là : Ta so sánh 2 vi tri gần nhau nếu chua đúng thứ tự , thì ta đổi cho cho nhau.

Sắp xếp chọn: là ta chọn trong mảng 1 số và tìm ra s. số nhỏ nhất trong mảng , sau đó đổi vi tri chúng đưa lên đầu. ko quan tâm đến nó nữa, bắt đau vi tri thứ 2, lặp lại quá trinh trên≥

Sắp xếp chèn: Nếu 2 phần tử canh nhau mà chưa đúng thư tự ta đổi vị tri cho nhau,

**Regex string:** regex là biểu thức chính quy, nhóm các ký tự, ký hiệu để tìm đoạn văn bản text.

(14-23) Cơ sở dữ liệu quan hệ: , Thiết kê CSDL, thao tác với dữ liệu, view, index, store procedure , câu lệnh CRUD, MVC mode. Css. Bootstrap

**Cở sở dữ liệu** là nhóm dữ liệu có cấu trúc, môt hình dữ liệu lưu trữ dữ liệu, quan lý dữ liệu, truy ván dữ liệu

**CSDLQH** là 1 cơ dư liêu dựa trên mối quan hệ , mối quan hệ giũa các bangr với nhau.

Mỗi bảng thì bao gồm cột dòng. Mỗi bảng chưa 1 thực thể duy nhát.a

**Hệ quan tri CSDL** là một phần mêm để quan lý cơ sở dữ liệu. CN đn nghia câp nhat truy ván, thêm sua xoa …

**Thiết kế CSDL:** b1. Xacs định muc đich của csdl . b2: tập hơp dư liêu tô chức cac bảng, khoa chinh ngoại. b3 tao mối liên hệ giua cac bảng. B4 tinh chỉnh chuẩn hoá

**Thao tác với dữ liệu:** 4 bảng join phổ biến, group by nhóm dữ liệu

**SQL view, index, store procedure.**

View là 1 bagnr ảo được tạo từ nhiều bảng gốc khac nhau. Muc đích để truy vấn vào những dữ liệu cần thiết nhát có trong 1 bảng ảo

CRUD: Create, tạo dữ lieu= INSERT, Read đọc dư liệu = SELECT, U = update, D = delete

Mo hình mvc là mô hình nhămf phân chia các đoạn code vào models, views, controllers, mục đihcs để chúng ta có thể dễ đọc, hiểu, bao tri, tái sử sd …

Một client gửi 1 yêu câu tới controllers thông qua điều hướng, controllers tiếp nhận xử lý gửi yêu cầu tới models để lấy dữ liệu về, controllers xử lý và gửi dữ liêu sang View thành HTML trả về controllers và hiển thị ra cho client.