**1. Interface là gì ?**

Interface là một Template (khuôn mẫu), nó không phải là một lớp đối tượng mà chỉ là một bề nhìn bên ngoài mà nhìn vào đó ta có thể biết được tất cả các hàm của đối tượng implement nó.

Để khai báo một Interface ta dùng từ khóa **interface** để thay cho từ khóa **class**. Tất cả các hàm trong interface đểu ở dạng khai báo và **không được định nghĩa** (giống lớp abstract). Nếu một đối tượng implement một interface thì nó phải khai báo và định nghĩa tất cả các hàm trong Interface.

Thoạt nhìn qua các bạn thấy Interface rất giống với Abstract trong php đúng không nào? Câu trả lời là bản chất bên trong hoàn toàn khác nhau. Interface không phải là một lớp cụ thể mà là một khuôn mẫu để cho **một** đối tượng implement nó, và đương nhiên là ta không thể tạo một biến Interface. Ngược lại [lớp Abstract](https://freetuts.net/lop-truu-tuong-abstract-trong-php-oop-37.html) là một lớp cụ thể, có đầy đủ các tính chất của một đối tượng, có thể gọi, định nghĩa các hàm trong nó. Đối với hằng số ở lớp implement không được định nghĩa lại.  
  
**Ví dụ**: Định nghĩa hằng số bị sai

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | interface A  {      const ConstA = 'Freetuts.net';  }    // Lớp này sai vì không thể định nghĩa lại hằng  class B implements A  {      const ConstA = 'Other Name';  }    // Lớp này đúng  class D implements A  {  } |

**Ví dụ**: Định nghĩa hàm trong template bị sai

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Khai báo một Interface  interface DogTemplate  {      // Hàm này đúng vì ta chỉ khai báo mà không có định nghĩa      public function Run();        // Hàm này sai vì ta đã định nghĩa cho nó      public function Eat(){        }  } |

**Ví dụ**: Định nghĩa mức truy cập bị sai

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | // Template  interface DogTemplate  {      public function Run();        public function Eat();  }    // Lớp Dog  class Dog implements DogTemplate  {      // Hàm này sai vì cấp độ truy cập      // của hàm run bên template là public      // mà trong hàm này ta lại khai báo là private      private function Run(){        }        // Hàm này đúng      public function Eat(){        }  } |

**Ví dụ**: Bị thiếu hàm

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | // Template  interface DogTemplate  {      public function Run();        public function Eat();  }    // Lớp Dog  // Sai vì thiếu hàm Run  class Dog implements DogTemplate  {      public function Eat(){        }  } |

**2.Tính kế thừa interface trong PHP**

**Interface trong php** tuy không phải là một [lớp](https://freetuts.net/lop-thuoc-tinh-phuong-thuc-cua-doi-tuong-28.html) chính hiệu nhưng nó cũng có một tính chất đó là [tính kế thừa](https://freetuts.net/tinh-ke-thua-trong-php-30.html), nghĩa là một Interface A có thể kế thừa một Interface B thì lúc này đối tượng nào implement lớp A thì nó phải định nghĩa tất cả các hàm mà cả hai lớp A và B đã khai báo.  
  
**Ví dụ:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | interface A {      public function funcA();  }    interface B extends A  {      public function funcB();  }    // Lớp này đúng vì nó khai báo đầy  // đủ các hàm trong A và B  class C implements B  {      public function funcA()      {        }        public function funcB()      {        }  }    // Lớp này sai vì nó khai báo mỗi hàm funcA  class D implements B  {      public function funcA()      {        }  } |