|  |  |
| --- | --- |
| **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.** | 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**. |