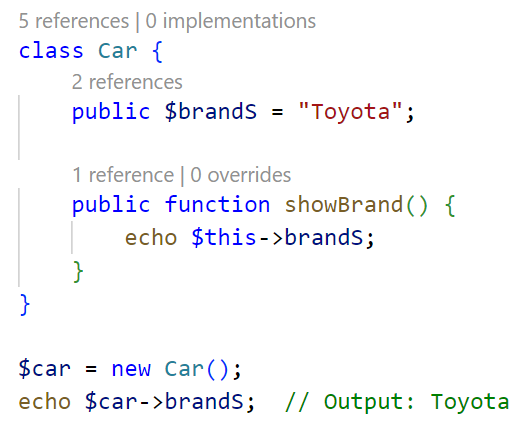
### 1. Phạm vi truy cập: public, protected và private; phương thức get và set

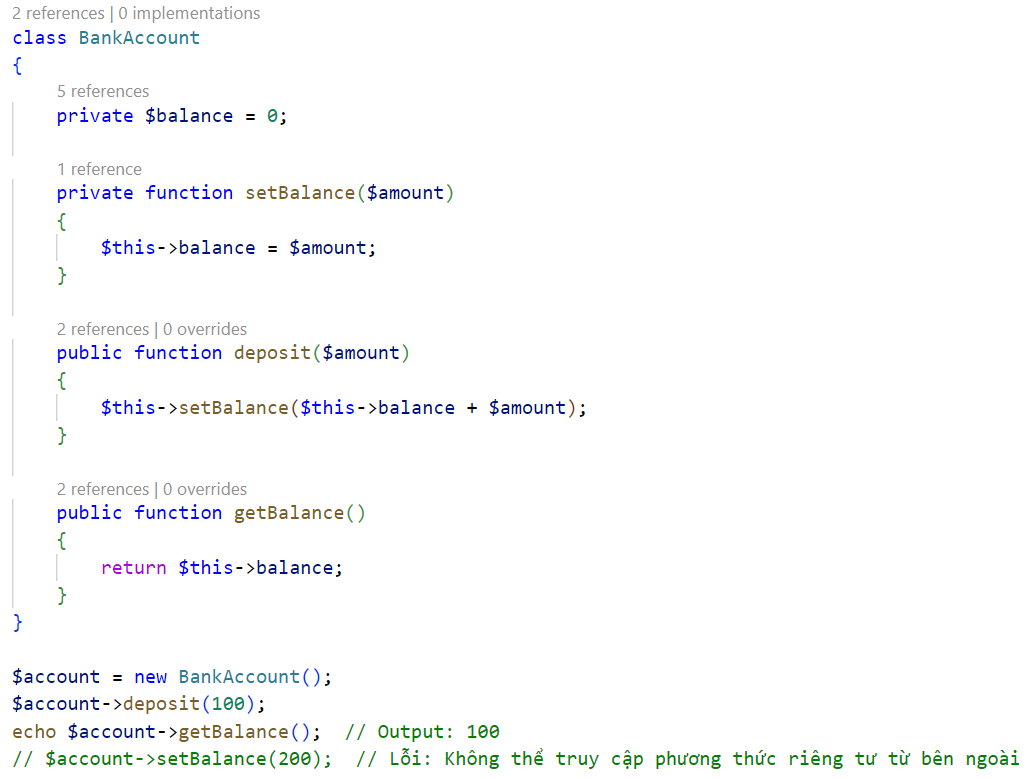
**Public (Công khai):** Thuộc tính hoặc phương thức được khai báo với phạm vi public có thể được truy cập từ bất kỳ đâu: từ bên trong lớp, lớp con, và từ bên ngoài lớp.



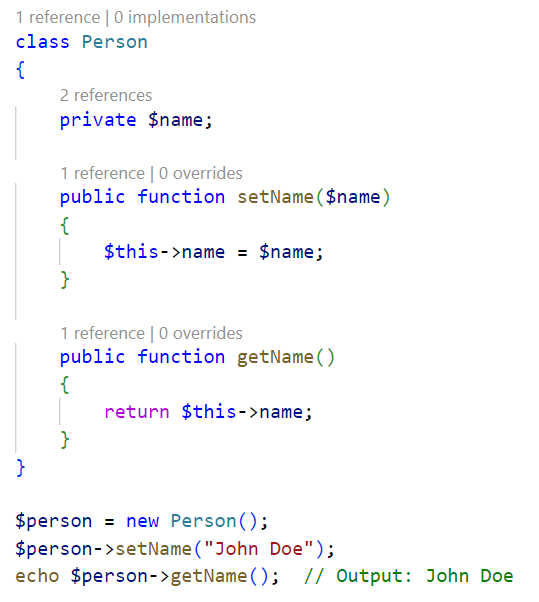
**Protected (Bảo vệ):** Thuộc tính hoặc phương thức được khai báo với phạm vi protected chỉ có thể được truy cập từ bên trong lớp đó và từ các lớp con (subclasses). Nó không thể được truy cập từ bên ngoài lớp.



**Private (Riêng tư):** Thuộc tính hoặc phương thức được khai báo với phạm vi private chỉ có thể được truy cập từ bên trong lớp đó. Nó không thể được truy cập từ các lớp con hoặc từ bên ngoài lớp.



****Phương thức** get **và** set:** Đây là các phương thức được sử dụng để truy cập và cập nhật giá trị của các thuộc tính có phạm vi bảo vệ (protected) hoặc riêng tư (private).



1. **Phân biệt Abstract và Interface**

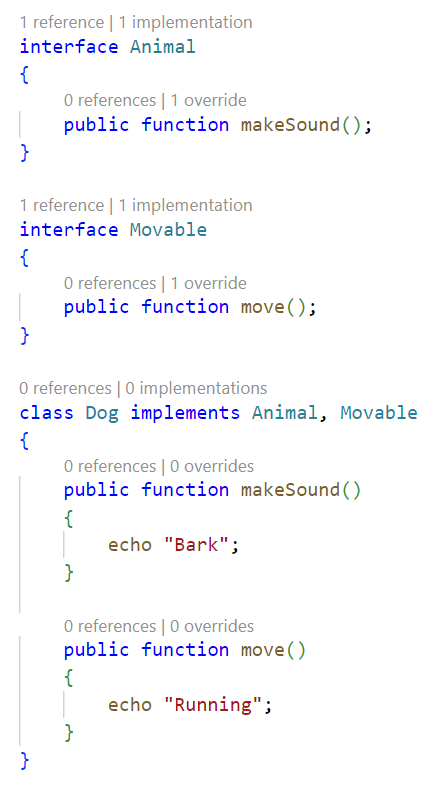
**Abstract (Lớp trừu tượng):**

* Một lớp trừu tượng không thể được khởi tạo trực tiếp; nó chỉ có thể được kế thừa.
* Lớp trừu tượng có thể chứa cả các phương thức trừu tượng (chưa được triển khai) và các phương thức đã được triển khai.
* Phương thức trừu tượng phải được các lớp con triển khai.



**Interface:**

* Interface là một hợp đồng mà các lớp phải tuân theo.
* Một interface chỉ chứa các phương thức trừu tượng (không có phần thân).
* Một lớp có thể triển khai (implement) nhiều interface, điều này giúp mô phỏng đa kế thừa trong PHP



**Sự khác biệt:**

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Abstract | Interface |
| Khai báo | abstract class ClassName | interface InterfaceName |
| Phương thức | Có thể chứa cả phương thức đã được triển khai và phương thức trừu tượng (không có phần thân). | Chỉ chứa các phương thức trừu tượng (không có phần thân). |
| Thuộc tính (Properties) | Có thể chứa thuộc tính (properties). | Không chứa thuộc tính, chỉ chứa các hằng số (constants). |
| Triển khai (Implementation) | Một lớp con phải sử dụng từ khóa extends để kế thừa lớp trừu tượng. | Một lớp phải sử dụng từ khóa implements để triển khai interface. |
| Số lượng kế thừa/triển khai | Một lớp chỉ có thể kế thừa từ một lớp trừu tượng. | Một lớp có thể triển khai nhiều interface. |
| Tính khả dụng của phương thức | Phương thức của lớp trừu tượng có thể có bất kỳ phạm vi truy cập nào (public, protected, private). | Phương thức của interface phải là public. |
| Tính trừu tượng | Có thể chứa cả phần thực hiện cụ thể và phần trừu tượng. | Hoàn toàn trừu tượng, không chứa phần thực hiện cụ thể. |
| Khi sử dụng | Sử dụng khi có một lớp cha cần có một số phương thức cụ thể mà mọi lớp con phải có, nhưng cũng cần có phương thức chung cho tất cả lớp con. | Sử dụng khi bạn cần đảm bảo rằng các lớp hoàn toàn khác nhau phải tuân theo một bộ phương thức chuẩn mà không có sự thực hiện mặc định. |

### 3. Đa kế thừa với Trait

PHP không hỗ trợ đa kế thừa trực tiếp (một lớp chỉ có thể kế thừa từ một lớp khác). Tuy nhiên, PHP cung cấp **Traits** để giải quyết vấn đề này. Trait cho phép bạn tái sử dụng các phương thức trong nhiều lớp mà không cần phải kế thừa.

* **Trait:** Là một nhóm phương thức có thể được sử dụng lại trong nhiều lớp khác nhau. Trait có thể bao gồm các phương thức và thuộc tính. Một lớp có thể sử dụng nhiều trait.



**Trait vs. Class:**

* Trait không thể được khởi tạo; nó chỉ được sử dụng để cung cấp các phương thức cho các lớp khác.
* Trait không phải là một lớp cha, do đó không có mối quan hệ kế thừa giữa trait và lớp sử dụng nó.