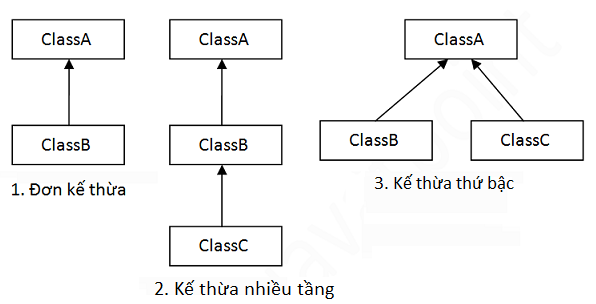
**Lập trình hướng đối tượng trong Java**

* Lập trình hướng đối tượng (Object-Oriented Programming) là 1 phương pháp hay mô hình giúp tăng năng suất, đơn giản hóa việc bảo trì, dễ dàng mở rộng trong thiết kế phần mềm
* Lập trình hướng đối tượng có 4 tính chất chính:
* Inheritance (Tính kế thừa)
* Encapsulation (Tính đóng gói)
* Abstraction (Tính trừu tượng)
* Polymorphism ( Tính đa hình)

1. **Tính kế thừa**

* Tính kế thừa trong Java là 1 subclass có thể thừa hưởng lại tất cả những phương thức và thuộc tính của superclass và có thể tự định nghĩa những phương thức và thuộc tính của riêng nó
* Phạm vi truy cập của subclass:
* Nếu cùng package, subclass không thể truy cập bất cứ thành viên được **private** của superclass
* Nếu khác package, subclass không thể truy cập bất cứ thành viên được **private và default** của superclass
* Có 3 kiểu kế thừa: **đơn kế thừa, kế thừa thứ bậc và kế thừa nhiều cấp** 

*Java không hỗ trợ đa kế thừa class vì để giảm thiểu sự phức tạp và đơn giản hóa ngôn ngữ*

*VD: class A kế thừa class B và class C, class B và class C có cùng 1 phương thức là D() -> class A bị mơ hồ khi không biết là gọi phương thức D() từ class B hay class C*

* *Để có thể đa kế thừa trong Java, chúng ta có thể đa kế thừa Interface*

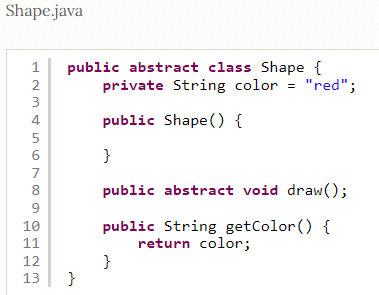
1. **Tính đóng gói**

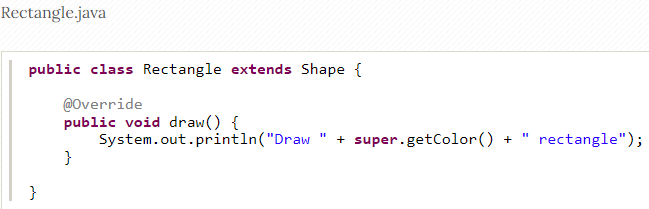
* Tính đóng gói là kỹ thuật ẩn giấu thông tin không liên quan và hiển thị ra thông tin liên quan , làm giảm thiểu mức độ phức tạp phát triển phần mềm
* Chúng ta có thể sử dụng setter và getter để thể hiện tính chất này

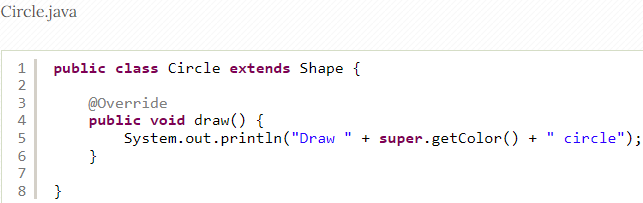
1. **Tính trừu tượng**

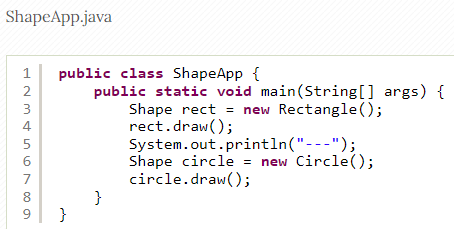
* Tính trừu tượng trong Java là 1 tiến trình ẩn chi tiết và chỉ hiển thị tính năng tới người dùng
* Thường dùng nhất sẽ là **abstract class** và **interface**
  1. ***Abstract class (lớp trừu tượng)***
* Một lớp được khai báo với từ khóa abstract là lớp trừu tượng
* Lớp trừu tượng có thể khai báo các phương thức abstract hoặc non-abstract
* Không thể khởi tạo 1 đối tượng trực tiếp từ 1 class trừu tượng
* Subclass kế thừa lớp trừu tượng bắt buộc phải override lại những abstract methods (trừ khi subclass cũng là abstract)

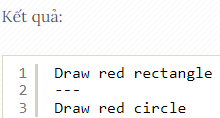
Ví dụ:









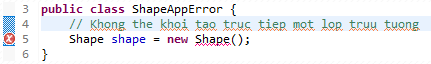


***Lưu ý****:*

* *Subclass bắt buộc phải implement lại tất cả các abstract method của superclass*

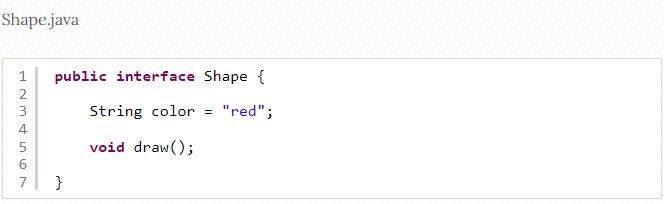
**

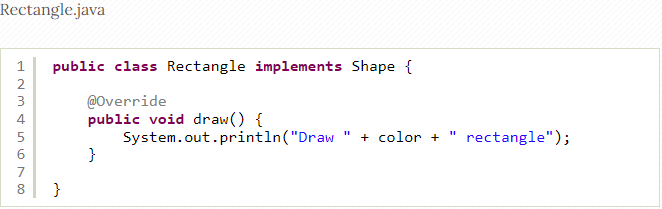
* *Compile error*
* *Không thể khởi tạo trực tiếp 1 abstract class*

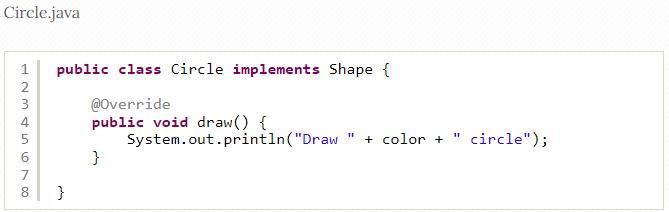
**

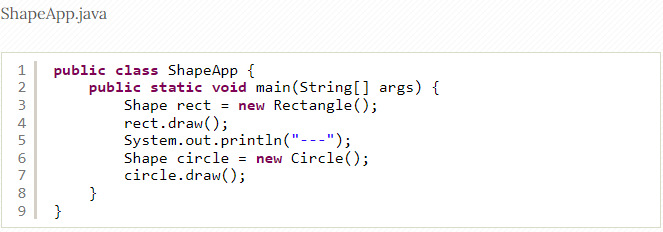
* *Compile error*
  1. ***Interface trong Java***
* Các phương thức trong interface đều là abstract methods
* Interface thể hiện được tính trừu tượng hoàn toàn và đa kế thừa trong Java
* Modifier của interface luôn là public interface
* Các fields trong interface luôn là public static final
* Các methods trong interface luôn là public abstract
* Interface không có constructor
* Class và interface khác nhau. Class mô tả thuộc tính và hành vi của đối tượng, trong khi interface chứa các hành vi mà 1 class triển khai
* Trừ khi lớp triển khai là lớp trừu tượng abstract, còn lại tất cả phương thức trong interface cần được định nghĩa trong class
* 1 interface có thể kế thừa từ interface khác, có thể đa kế thừa interface
* 1 class có thể triển khai được nhiều interface
* 1 interface có thể được lồng trong 1 interface khác

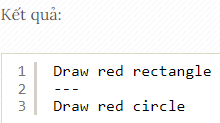
Ví dụ:











* 1. ***So sánh abstract class và interface***

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| Có các phương thức abstract và non-abstract | Chỉ có các phương thức abstract |
| Không hỗ trợ đa kế thừa | Hỗ trợ đa kế thừa |
| Có thể có các biến final, non-final, static, non-static | Chỉ có biến static final |
| Có thể có các phương thức static, main và có constructor | Không thể có các phương thức static, main và constructor |
| Có thể cung cấp trình triển khai interface | Interface không cung cấp trình triển khai cụ thể của lớp abstract |

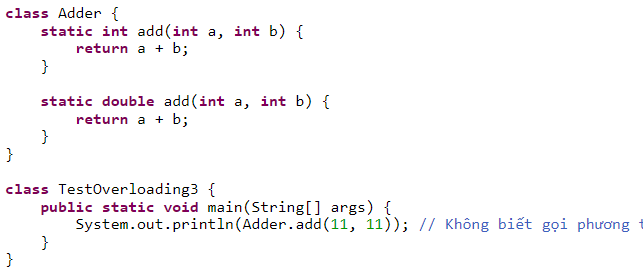
1. **Tính đa hình**

Tính đa hình là 1 nhiệm vụ được triển khai theo nhiều cách khác nhau. Trong Java để thực hiện tính đa hình, chúng ta thường sử dụng overloading (nạp chồng) và overriding(ghi đè)

* **Nạp chồng (overloading):** Overloading hay còn gọi là tính đa hình khi compile. Cho phép 1 lớp có nhiều thuộc tính, phương thức cùng tên nhưng với các tham số khác nhau về loại cũng như về số lượng hoặc khác kiểu dữ liệu trả về
* **Ghi đè (overriding):** Overriding hay còn gọi là tính đa hình khi runtime. 2 phương thức cùng tên, cùng tham số, cùng kiểu trả về nhưng subclass viết lại và dùng theo cách của nó, và xuất hiện ở lớp cha và tiếp tục xuất hiện ở lớp con

***Câu hỏi về ghi đè:***

* ***Có ghi đè được phương thức static không? Vì sao?***
* *Không thể ghi đè phương thức static. Vì ghi đè dựa vào quá trình gán động lúc runtime, trong khi phương thức static được gán tĩnh lúc compile*
* ***Có thể ghi đè phương thức main được không?***
* *Không, vì phương thức main là static*
* ***Có thể nạp chồng chỉ bằng cách thay đổi kiểu dữ liệu trả về của phương thức hay không? Vì sao****?*
* *Không, vì nếu chỉ thay đổi kiểu trả về của phương thức thì sẽ không biết phương thức nào sẽ được gọi*

******