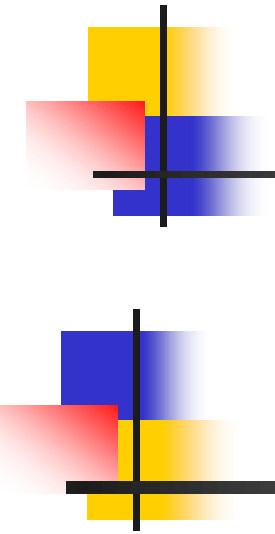
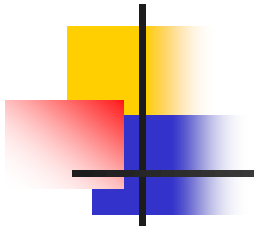


CÁC TÍNH CHẤT CỦA OOP TRONG JAVA

Tính Đóng Gói (Encapsulation)

Tính Trừu Tượng (Abstract)

Interface Trong Java

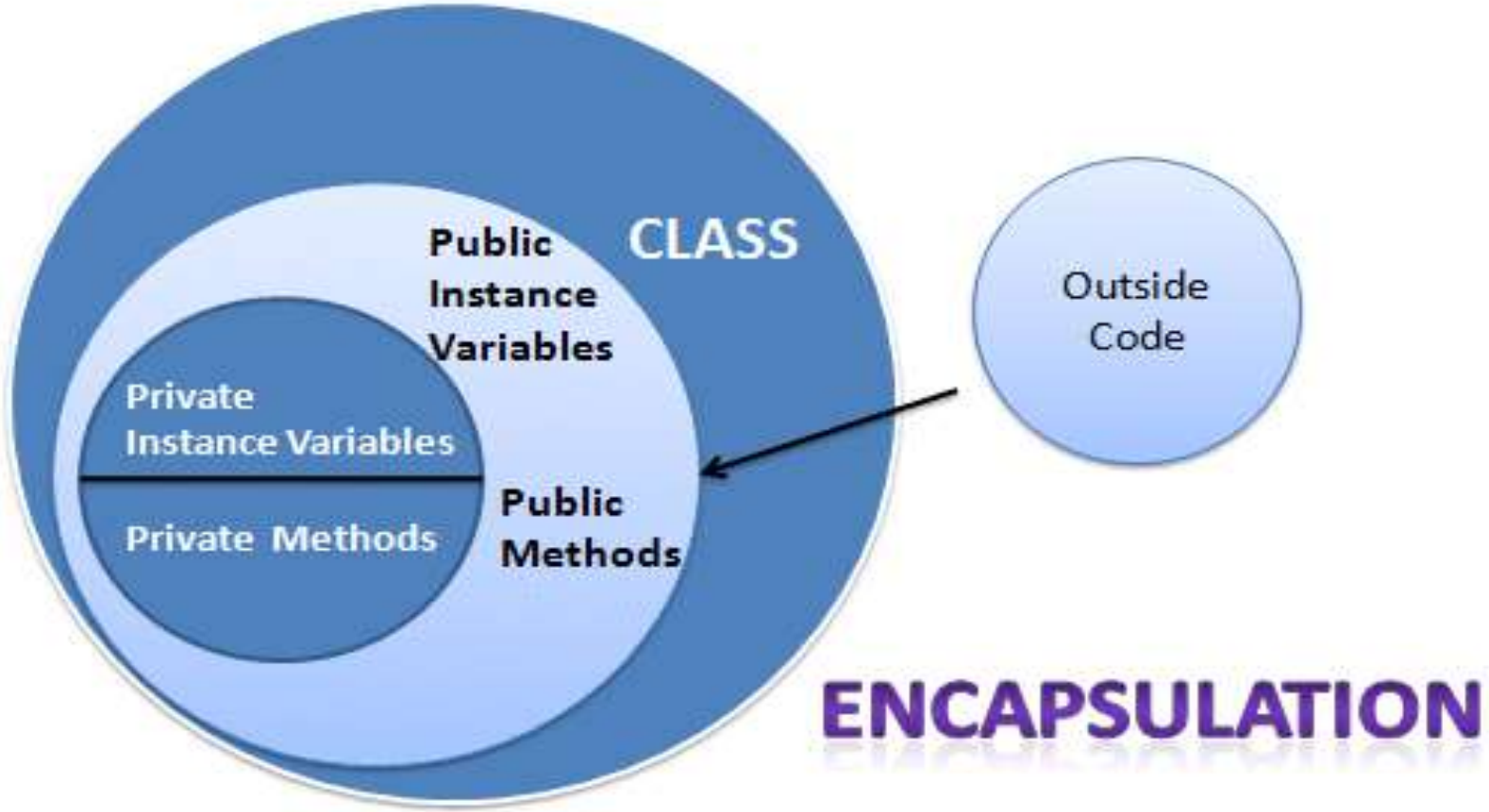


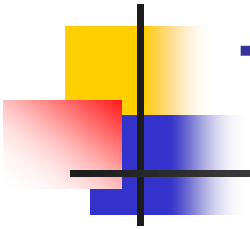
Tính Đóng Gói (Encapsulation)

Tính Đóng Gói

- Tính đóng gói trong java là kỹ thuật che giấu những chi tiết hiện thực của đối tượng (kỹ thuật ẩn giấu thông tin không liên quan và hiển thị ra thông tin cần thiết có liên quan đối tượng)
- Mục đích chính của đóng gói trong java là giảm thiểu mức độ phức tạp phát triển phần mềm.

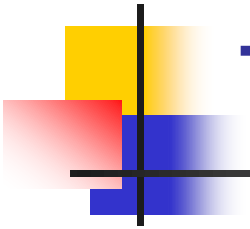
Tính Đóng Gói





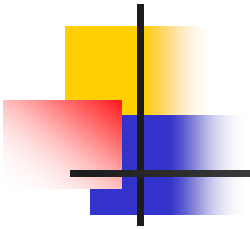
Tính Đóng Gói

- Đóng gói cũng được sử dụng để bảo vệ trạng thái bên trong của một đối tượng. Bởi việc ẩn giấu các biến biểu diễn trạng thái của đối tượng.
- Việc chỉnh sửa đối tượng được thực hiện, xác nhận thông qua các phương thức.
- Việc ẩn giấu các biến thì các lớp sẽ không chia sẻ thông tin với nhau được. Điều này làm giảm số lượng khớp nối có thể có trong một ứng dụng.

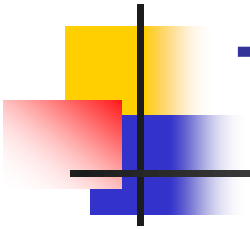


Tính Đóng Gói

- Ta có thể tạo lớp read-only hoặc write-only bằng việc cài đặt phương thức setter hoặc getter.
- Ta có thể kiểm soát đối với dữ liệu.
- Ví dụ: *Student.java, Test.java*



Tính Trừu Tượng (Abstract)

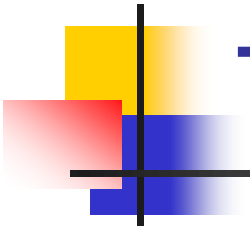


Tính Trừu Tượng

- Một lớp được khai báo với từ khóa abstract là lớp abstract trong Java.
- Lớp abstract có nghĩa là lớp trừu tượng
  - Lớp abstract có thể có các phương thức abstract hoặc phương thức non-abstract
- Trước khi tìm hiểu về lớp trừu tượng trong Java, ta cần hiểu tính trừu tượng trong Java là gì?

Tính Trừu Tượng

- Tính trừu tượng là một tiến trình ẩn các cài đặt chi tiết và chỉ hiển thị tính năng tới người dùng.
- Nói cách khác, nó chỉ hiển thị các thứ quan trọng tới người dùng và ẩn các chi tiết nội tại
- Ví dụ: Để gửi tin nhắn, người dùng chỉ cần soạn tin nhắn và gửi đi. Ta không cần biết tiến trình xử lý nội tại về phân phối tin nhắn.
- Tính trừu tượng giúp bạn trọng tâm hơn vào đối tượng thay vì quan tâm đến cách nó thực hiện.



Tính Trừu Tượng

- Các cách để đạt được sự trừu tượng hóa
  - Sử dụng lớp abstract
  - Sử dụng interface

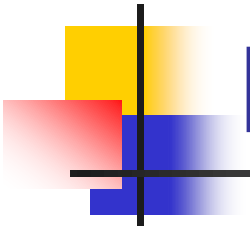
Phương Thức Trừu Tượng

- Một phương thức được khai báo là abstract và không có trình triển khai (body) thì đó là phương thức trừu tượng.
- Từ khóa abstract được sử dụng để khai báo một phương thức dạng abstract.
- Một phương thức abstract không có thân phương thức.

Phương Thức Trừu Tượng

- Phương thức abstract sẽ không có định nghĩa, được theo sau bởi dấu chấm phảy, không có dấu ngoặc theo sau

```
/* Khai báo phương thức với từ khóa abstract và không có thân phương thức */
abstract void InitMethod();
```



Phương Thức Trừu Tượng

- Nếu ta muốn một lớp chứa một phương thức cụ thể và ta muốn triển khai thực sự phương thức đó để được quyết định bởi các lớp con
- Thì ta có thể khai báo phương thức trong lớp cha ở dạng abstract.

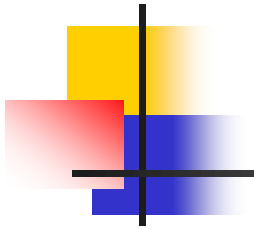
Phương Thức Trừu Tượng

- Ví dụ về lớp trừu tượng và phương thức trừu tượng trong Java
  - Bike là lớp trừu tượng chỉ chứa một phương thức trừu tượng là run().
  - Trình triển khai của nó được cung cấp bởi lớp Motor

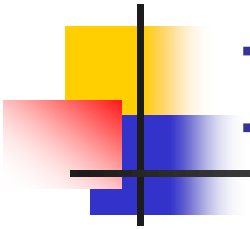
(Motor.java)

- Ví dụ về kế thừa lớp Abstract trong Java
  - Shape là lớp trừu tượng, trình triển khai của nó được cung cấp bởi lớp Rectangle và lớp Circle.
  - Hai lớp này kế thừa lớp trừu tượng Shape.

(TestAbstract.java)

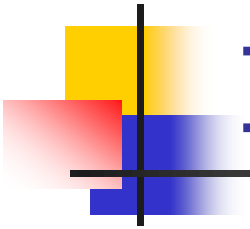


Interface Trong Java



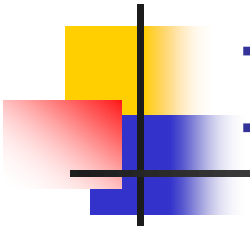
Interface Trong Java

- Interface là lớp trừu tượng thuần tuý.
- Một Interface trong Java là một tập hợp các phương thức trừu tượng (abstract).
- *Interface là một kỹ thuật để thu được tính trừu tượng hoàn toàn và đa kế thừa trong Java*.



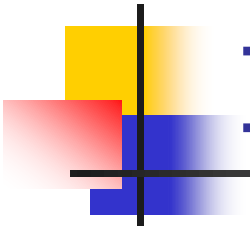
Interface Trong Java

- Java Compiler thêm từ khóa public và abstract trước phương thức của interface và các từ khóa public, static và final trước các thành viên dữ liệu.
- Nói cách khác, các trường của Interface là public, static và final theo mặc định và các phương thức là public và abstract.
- Một class triển khai một interface, do đó kế thừa các phương thức abstract của interface.



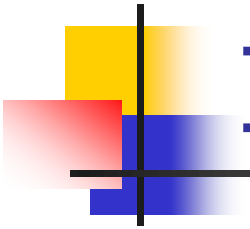
Interface Trong Java

- Một interface không phải là một lớp. Viết một interface giống như viết một lớp, nhưng chúng có hai định nghĩa khác nhau.
- Một lớp mô tả các thuộc tính và hành vi của một đối tượng.
- Một interface chứa các hành vi mà một class triển khai.
- Trừ khi một lớp triển khai interface là lớp trừu tượng abstract, còn lại tất cả các phương thức của interface cần được định nghĩa trong class.



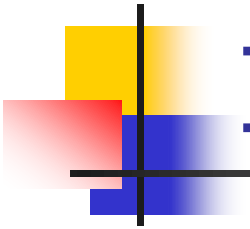
Interface Trong Java

- Một interface tương tự với một class bởi những điểm sau đây:
  - Một interface được viết trong một file với định dạng *.java, với tên của interface giống tên của file
  - Bytecode của interface được lưu trong file có định dạng *.class
  - Khai báo interface trong một package, những file bytecode tương ứng cũng có cấu trúc thư mục có cùng tên package



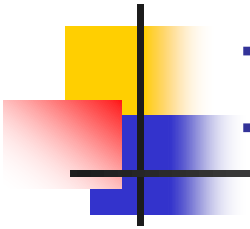
Interface Trong Java

- Một interface khác với một class ở một số điểm sau đây:
  - Ta không thể khởi tạo một interface.
  - Một interface không chứa bất cứ hàm Constructor nào.
  - Tất cả các phương thức của interface đều là abstract.
  - Một interface không thể chứa một trường nào trừ các trường vừa static và final.
  - Một interface không thể kế thừa từ lớp, nó được triển khai bởi một lớp.
  - Một interface có thể kế thừa từ nhiều interface khác.
- Ví dụ: *A.java*



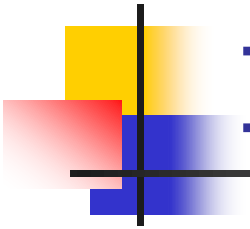
Interface Trong Java

- Khi triển khai interface, có vài quy tắc sau:
  - Một lớp có thể triển khai một hoặc nhiều interface tại một thời điểm.
  - Một lớp chỉ có thể kế thừa một lớp khác, nhưng được triển khai nhiều interface.
  - Một interface có thể kế thừa từ một interface khác, tương tự cách một lớp có thể kế thừa lớp khác.



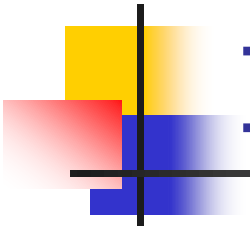
Interface Trong Java

- Đa kế thừa trong Java bởi Interface
  - Nếu một lớp triển khai đa kế thừa, hoặc một Interface kế thừa từ nhiều Interface thì đó là đa kế thừa *(A1.java)*
- Đa kế thừa không được hỗ trợ thông qua lớp trong Java nhưng là có thể bởi Interface
  - Trong nội dung về tính kế thừa, đa kế thừa không được hỗ trợ thông qua lớp.
  - Nó được hỗ trợ bởi Interface bởi vì không có tính lưỡng nghĩa khi trình triển khai được cung cấp bởi lớp Implementation *(TestInterface.java)*



Interface Trong Java

- Kế thừa Interface trong Java
  - Một lớp triển khai Interface nhưng một Interface kế thừa từ Interface khác *(TestInterface1.java)*
- Marker(Tagging) Interface trong Java
  - Đó là một Interface mà không có thành viên nào.
  - Ví dụ: Serializable, Cloneable, Remote, …
  - Chúng được sử dụng để cung cấp một số thông tin thiết yếu tới JVM để mà JVM có thể thực hiện một số hoạt động hữu ích.



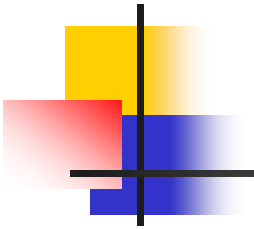
Interface Trong Java

- Có hai mục đích thiết kế chủ yếu của tagging interface là:
  - Thêm một kiểu dữ liệu tới một class: Đó là khái niệm tagging
  - Một class mà triển khai một tagging interface không cần định nghĩa bất kỳ phương thức nào
  - Nhưng class trở thành một kiểu interface thông qua tính đa hình

Interface Trong Java

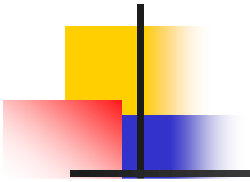
- Interface lồng nhau trong Java
  - Một Interface có thể có Interface khác, đó là lồng Interface. Chúng ta sẽ tìm hiểu chi tiết về lồng các lớp trong Java
  - Ví dụ:

```
interface printable{
        void print();
        interface MessagePrintable{
        void msg();
        }
}
```



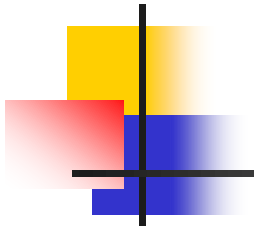
Sự khác nhau giữa Abstract class và Interface

- Abstract class và interface đều được sử dụng để có được sự trừu tượng mà ở đó chúng ta có thể khai báo các phương thức trừu tượng.
- Nhưng có một vài sự khác nhau giữa abstract class và interface



Sự khác nhau giữa Abstract class và Interface

1. Abstract class có phương thức abstract (không có thân hàm) và phương thức non-abstract (có thân hàm)	Interface chỉ có phương thức abstract. Từ Java 8 có thêm phương thức default và static
2. Abstract class không hỗ trợ đa kế thừa	Interface có hỗ trợ đa kế thừa
3. Abstract class có các biến final, non-final, static và non-static	Interface chỉ có các biến static và final
4. Abstract class có thể cung cấp nội dung cài đặt cho phương thức của interface	Interface không cung cấp nội dung cài đặt cho phương thức abstract class
5. Từ khóa abstract được dùng để khai báo abstract class	Từ khóa interface dùng để khai báo interface



Hết!!!