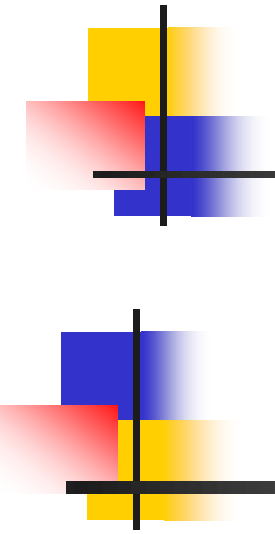


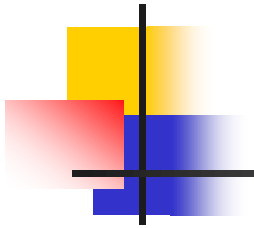
CÁC TÍNH CHẤT CỦA OOP TRONG JAVA

Tính Kế Thừa(Inheritance)

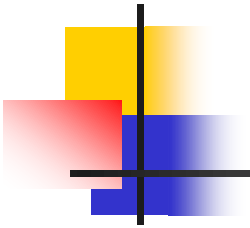
Tính Đa Hình(Polymorphism)

Ghi Đè Phương Thức(Method Overriding)

Nạp Chồng Phương Thức (Method Overloading)



Tính Kế Thừa (Inheritance)



Tính Kế Thừa

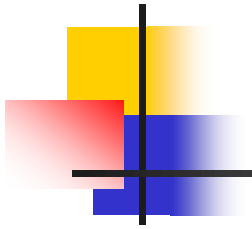
- Kế thừa trong java là sự liên quan giữa hai class với nhau, trong đó có class cha (superclass) và class con (subclass).
- Khi kế thừa class con được hưởng tất cả các phương thức và thuộc tính của class cha.
- Tuy nhiên, nó chỉ được truy cập các thành viên public và protected của class cha.
- Nó không được phép truy cập đến thành viên private của class cha.

Tính Kế Thừa

- Tư tưởng kế thừa trong java là có thể tạo ra một class mới được xây dựng trên các lớp đang tồn tại.
- Khi kế thừa từ một lớp đang tồn tại ta có thể sử dụng lại các phương thức và thuộc tính của lớp cha
- Đồng thời có thể khai báo thêm các phương thức và thuộc tính khác.
- Sử dụng từ khóa *extends* để kế thừa *(InheritanceSample1.java)*

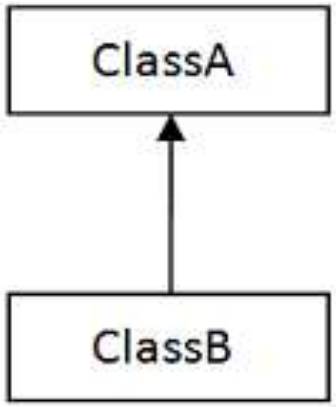
```
class Subclass_name extends Superclass_name {
}
```

Các kiểu kế thừa trong java

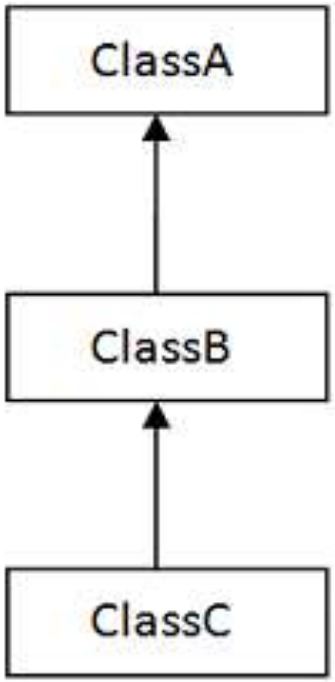


- Có 3 kiểu kế thừa trong java đó là:
  - Đơn kế thừa (TestInheritance1.java)
  - Kế thừa nhiều cấp (TestInheritance2.java)
  - Kế thừa thứ bậc (TestInheritance3.java)
- Khi một class được kế thừa từ nhiều lớp được gọi là đa kế thừa.
- Trong java, đa kế thừa chỉ được hỗ trợ thông qua interface
- Đa kế thừa trong java không được hỗ trợ thông qua class.

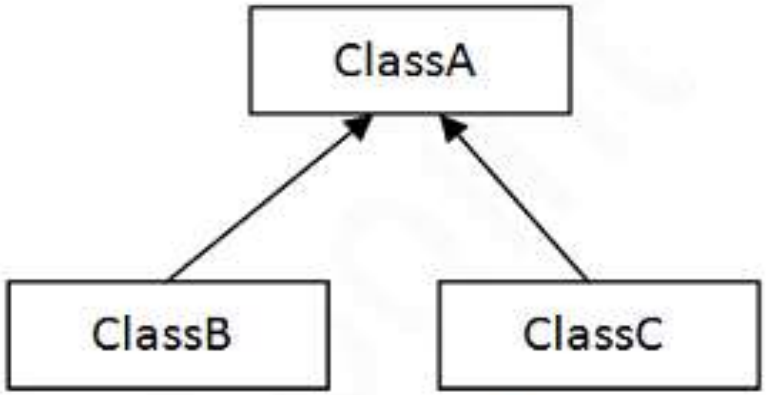
Các kiểu kế thừa trong java



1. Đơn kế thừa

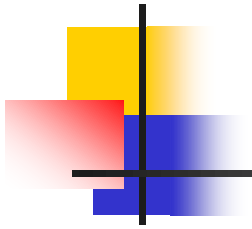


2. Kế thừa nhiều tầng



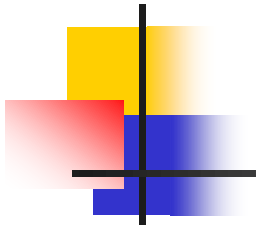
3. Kế thừa thứ bậc

Đa kế thừa không được hỗ trợ trong java

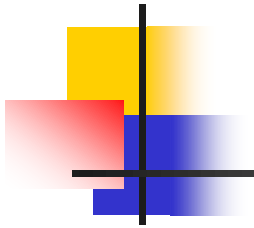


- Để giảm thiểu sự phức tạp và đơn giản hóa ngôn ngữ, đa kế thừa không được hỗ trợ trong java.
- Có 3 lớp A, B, C. Trong đó lớp C kế thừa từ các lớp A và B. Nếu các lớp A và B có phương thức giống nhau và ta gọi nó từ đối tượng của lớp con

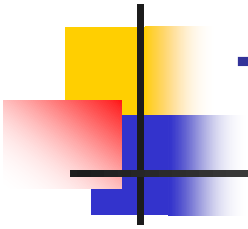
Đa kế thừa không được support trong java



- Như vậy khó có thể xác đinh được việc gọi phương thức của lớp A hay B.
- Vì vậy lỗi khi biên dịch sẽ tốt hơn lỗi khi runtime, java sẽ in ra lỗi "compile time error" nếu ta cố tình kế thừa hai class (C.java)

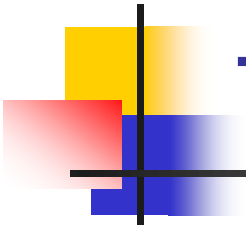


Tính Đa Hình (Polymorphism)



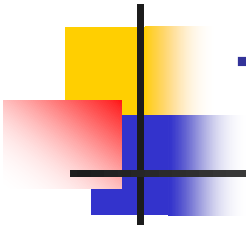
Tính Đa Hình

- Đa hình trong java (Polymorphism) là một khái niệm mà chúng ta có thể thực hiện một hành động bằng nhiều cách khác nhau
- Đa hình lúc biên dịch (compile) và đa hình lúc thông dịch (runtime).
- Chúng ta có thể thực hiện đa hình trong java bằng cách nạp chồng phương thức và ghi đè phương thức.



Tính Đa Hình

- Nếu ta nạp chồng phương thức static trong java, đó là đa hình lúc biên dịch
- Đa hình lúc thông dịch (runtime) là quá trình gọi phương thức đã được ghi đè trong thời gian thực thi chương trình.
- Trong quá trình này, một phương thức được ghi đè được gọi thông qua biến tham chiếu của một lớp cha

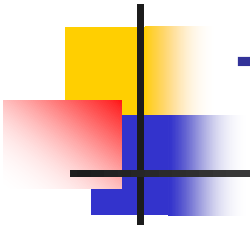


Tính Đa Hình

- Trước khi tìm hiểu về đa hình tại runtime, chúng ta cùng tìm hiểu về Upcasting
- Khi biến tham chiếu của lớp cha tham chiếu tới đối tượng của lớp con, thì đó là Upcasting
- Ví dụ:

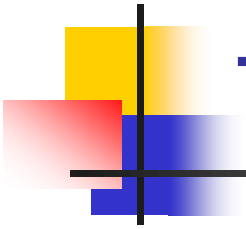
```
class    A{}
class  B    extends   A{}
A   a= new  B();// Upcasting
```

// Motor.java, Bank.java



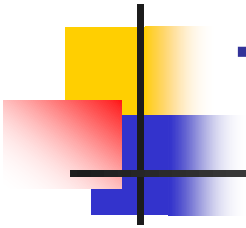
Tính Đa Hình

- Đa hình lúc runtime trong Java với thành viên dữ liệu
- Phương thức bị ghi đè không là thành viên dữ liệu
- *Vì thế đa hình lúc thông dịch (runtime) không thể có được bởi thành viên dữ liệu.*



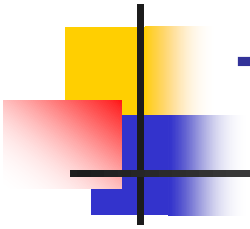
Tính Đa Hình

- Trong ví dụ sau đây, cả hai lớp có một thành viên dữ liệu là *speed*
- Chúng ta truy cập thành viên dữ liệu bởi biến tham chiếu của lớp cha mà tham chiếu tới đối tượng lớp con (Upcasting)
- Khi chúng ta truy cập *thành viên dữ liệu* mà không bị ghi đè, thì nó sẽ luôn luôn truy cập thành viên dữ liệu của lớp cha
- *Chú ý là đa hình lúc runtime không thể xảy ra với thành viên dữ liệu (Harley1.java)*



Tính Đa Hình

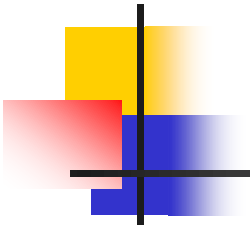
```
class  Motor{
   int    speed=80;
  }
 class HarLey extends Motor{
     int   speed=120;
   public static void main(String args[]){
      Motor obj=new Harley();  //upcasting
      System.out.println(obj.speed);
  }
```



Tính Đa Hình

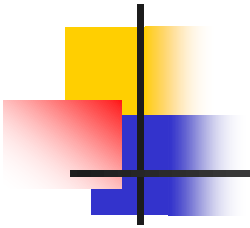
- Đa hình lúc thông dịch (runtime) trong Java với kế thừa nhiều tầng
- *Ví dụ : BabyCat.java và BabyCat1.java*

Nạp chồng phương thức (method overloading)



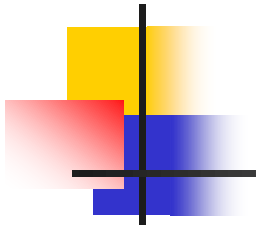
- Nạp chồng phương thức trong java xảy ra nếu một lớp có nhiều phương thức có tên giống nhau nhưng khác nhau về kiểu dữ liệu hoặc số lượng các tham số.
- Có 2 cách để nạp chồng phương thức trong java
  - Thay đổi số lượng các tham số
  - Thay đổi kiểu dữ liệu của các tham số

Nạp chồng phương thức (method overloading)



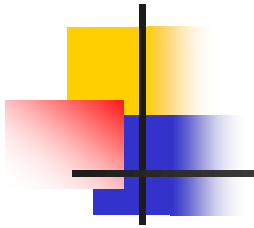
- Lợi ích của nạp chồng phương thức là tăng khả năng đọc hiểu chương trình.
- Trong java, *không thể nạp chồng phương thức bằng cách chỉ thay đổi kiểu trả về của phương thức.*

Nạp chồng phương thức (method overloading)



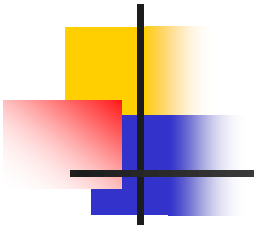
- Nạp chồng phương thức bằng cách thay đổi số lượng các tham số
- Chúng ta cần tạo 2 phương thức
  - phương thức add() đầu tiên thực hiện việc tính tổng của 2 số
  - phương thức thứ hai thực hiện việc tính tổng của 3 số.
  - Sử dụng phương thức static để gọi hàm thông qua tên class thay vì phải tạo thể hiên của lớp.

Nạp chồng phương thức (method overloading)



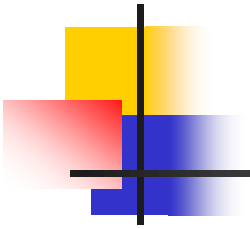
- Nạp chồng phương thức bằng cách thay đổi kiểu dữ liệu của các tham số
- Chúng ta sẽ tạo ra 2 phương thức có kiểu dữ liệu khác nhau.
  - Phương thức add() đầu tiên nhận 2 đổi số có kiểu giá trị là integer
  - Phương thức thứ hai nhận 2 đổi số có kiểu giá trị là double.

Nạp chồng phương thức (method overloading)



- Trong java, không thể nạp chồng phương thức bằng cách chỉ thay đổi kiểu trả về của phương thức bởi vì không biết phương thức nào sẽ được gọi (TestOverload3)
- Ta có thể nạp chồng n phương thức main. Nhưng JVM chỉ gọi phương thức main() có tham số truyền vào là một mảng String (TestOverload4)

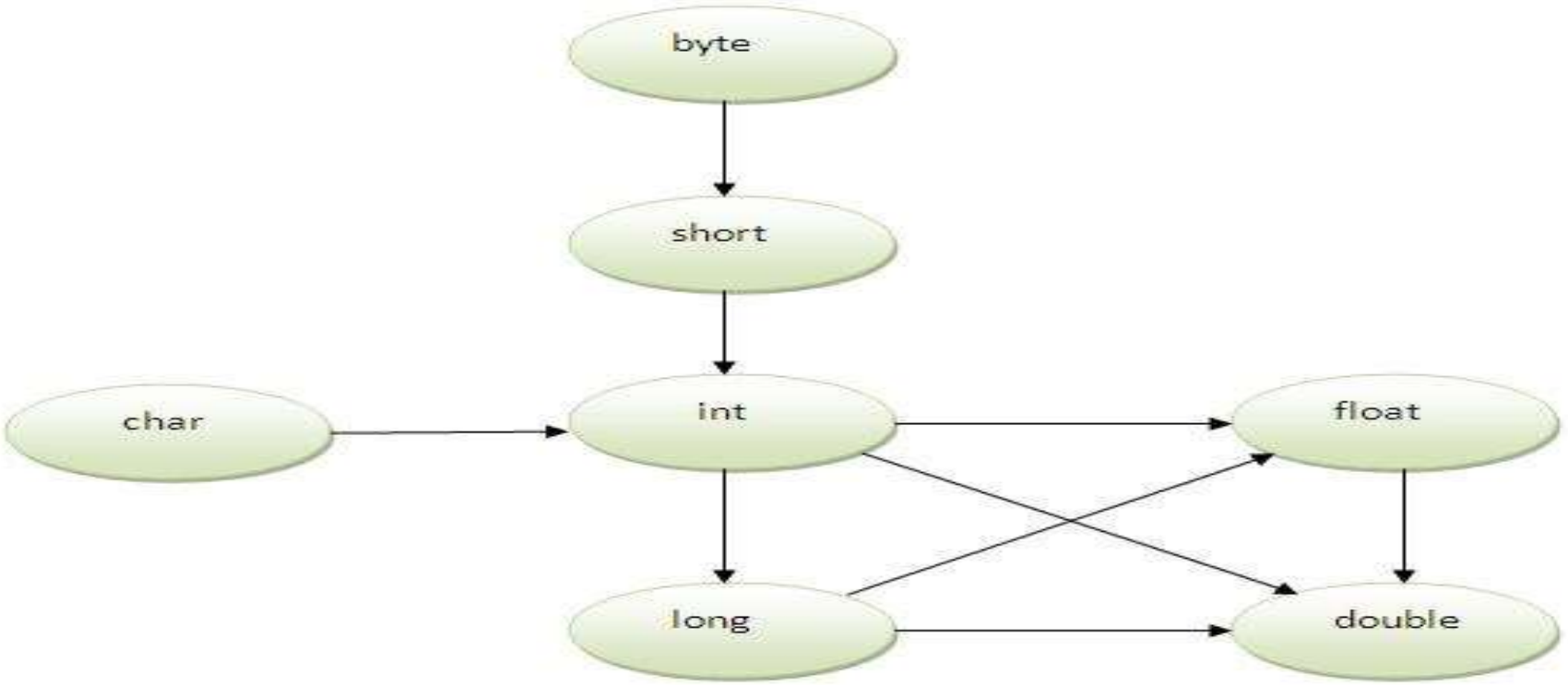
Nạp chồng phương thức (method overloading)



- Nạp chồng phương thức và sự thay đổi kiểu giá trị
  - Kiểu dữ liệu của đối số truyền vào được thay đổi sang kiểu dữ liệu khác (tự động ép kiểu)
  - Nếu giá trị của đối số đó không phù hợp với kiểu dữ liệu của tham số đã được định nghĩa.

Nạp chồng phương thức (method overloading)

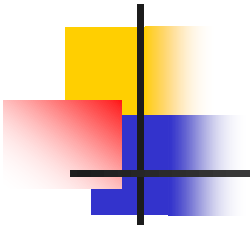
- Nạp chồng phương thức và sự thay đổi kiểu giá trị



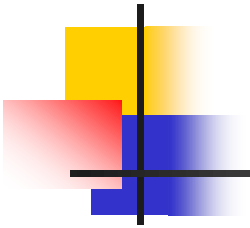
Nạp chồng phương thức (method overloading)

- Nạp chồng phương thức và sự thay đổi kiểu giá trị
  - Kiểu byte có thể được ép sang các kiểu short, int, long, float hoặc double.
  - Kiểu dữ liệu short có thể được ép sang các kiểu int, long, float hoặc double.
  - Kiểu dữ liệu char có thể được ép sang các kiểu int, long, float or double…
  - nếu không có kiểu đối số nào phù hợp, chuyển đổi kiểu sẽ không được thực hiện.

Nạp chồng phương thức (Method Overloading)

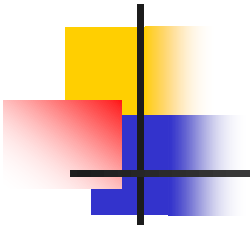


- Nạp chồng phương thức và sự thay đổi kiểu giá trị
  - Nếu không có kiểu đối số nào phù hợp, chuyển đổi kiểu sẽ không được thực hiện (OverloadingCal2)
  - Không có kiểu đối số nào phù hợp trong phương thức và mỗi phương thức thay đổi số đối số tương tự nhau. Trường hợp này sẽ không xác định được phương thức nào được gọi (OverloadingCal3)
  - Một kiểu không được tự động ép sang kiểu bé hơn, ví dụ kiểu double không được ép sang bất kỳ kiểu nào khác.



Ghi Đè Phương Thức (Method Overriding)

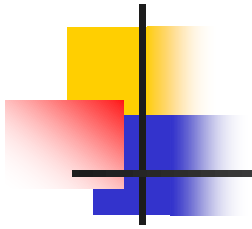
- Ghi đè phương thức trong java xảy ra nếu lớp con có phương thức giống lớp cha.
- Nói cách khác, nếu lớp con cung cấp sự cài đặt cụ thể cho phương thức đã được cung cấp bởi một lớp cha của nó được gọi là ghi đè phương thức (method overriding) trong java.



Ghi Đè Phương Thức (method overriding)

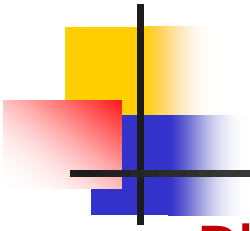
- Sử dụng ghi đè phương thức trong java
  - Ghi đè phương thức được sử dụng để cung cấp cài đặt đặc biệt của một phương thức mà đã được định nghĩa ở lớp cha.
  - Ghi đè phương thức được sử dụng cho đa hình runtime.

Ghi Đè Phương Thức (method overriding)



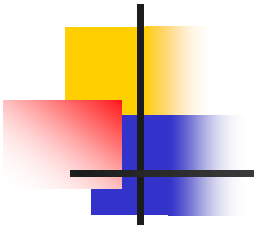
- Các nguyên tắc ghi đè phương thức trong java
  - Phương thức phải có tên giống với lớp cha.
  - Phương thức phải có tham số giống với lớp cha.
  - Lớp con và lớp cha có mối quan hệ kế thừa.

Ghi Đè Phương Thức (method overriding)



- Phương thức static không thể ghi đè được, bằng chứng là đa hình runtime
- Phương thức static được ràng buộc với class còn phương thức instance được ràng buộc với đối tượng.
  - *static thuộc về vùng nhớ class còn instance thuộc về vùng nhớ heap.*
- Có ghi đè phương thức main được không?
  - Không, vì main là phương thức static.

Khác nhau giữa overloading và overriding



Overloading

- Nạp chồng phương thức giúp code chương trình dễ đọc hơn
- Được thực hiện bên trong một class
- Tham số phải khác nhau
- Đa hình lúc biên dịch
- Kiểu giá trị trả về có thể giống hoặc khác nhau nhưng tham số phải khác nhau

Overriding

- Được sử dụng để cung cấp cài đặt cụ thể cho phương thức được khai báo của lớp cha
- Xảy ra trong 2 class có quan hệ kế thừa
- Tham số phải giống nhau
- Đa hình lúc runtime
- Giá trị trả về phải giống nhau