NÂNG CAO OOP VỚI JAVA

Người hướng dẫn: DiệuNT1

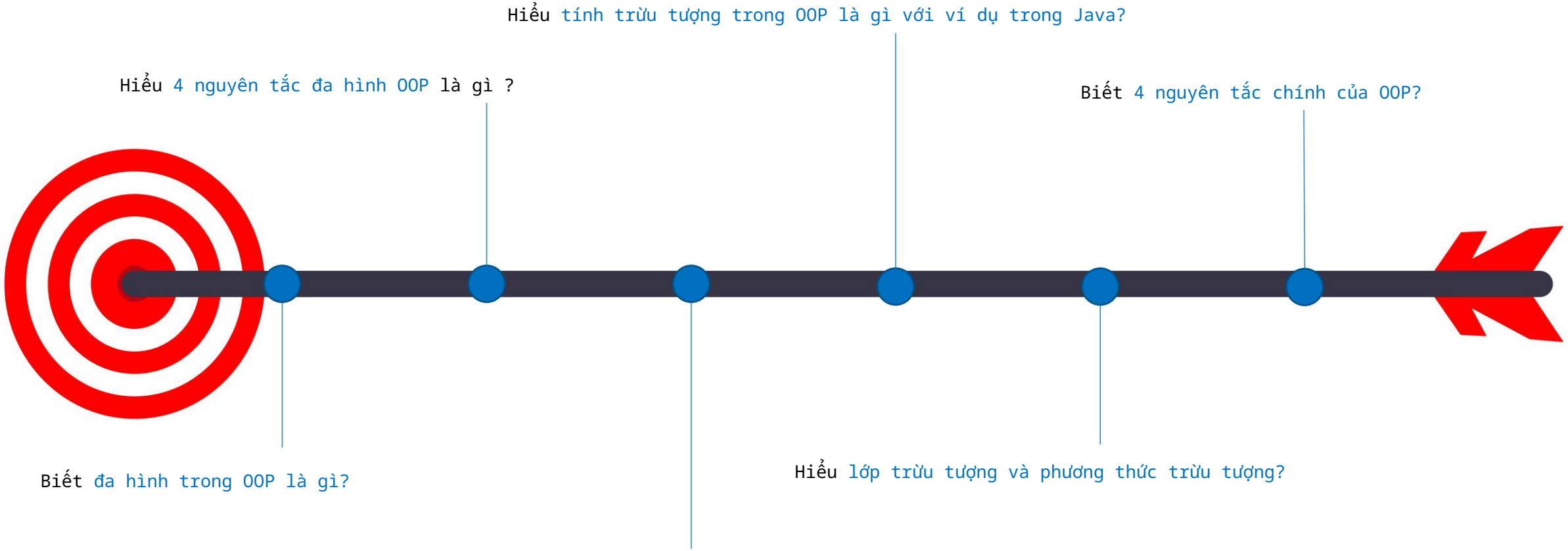
01.Tính đa hình

- Các dạng đa hình
- Quá tải phương thức
- Ghi đè phương thức
- Liên kết tĩnh và động

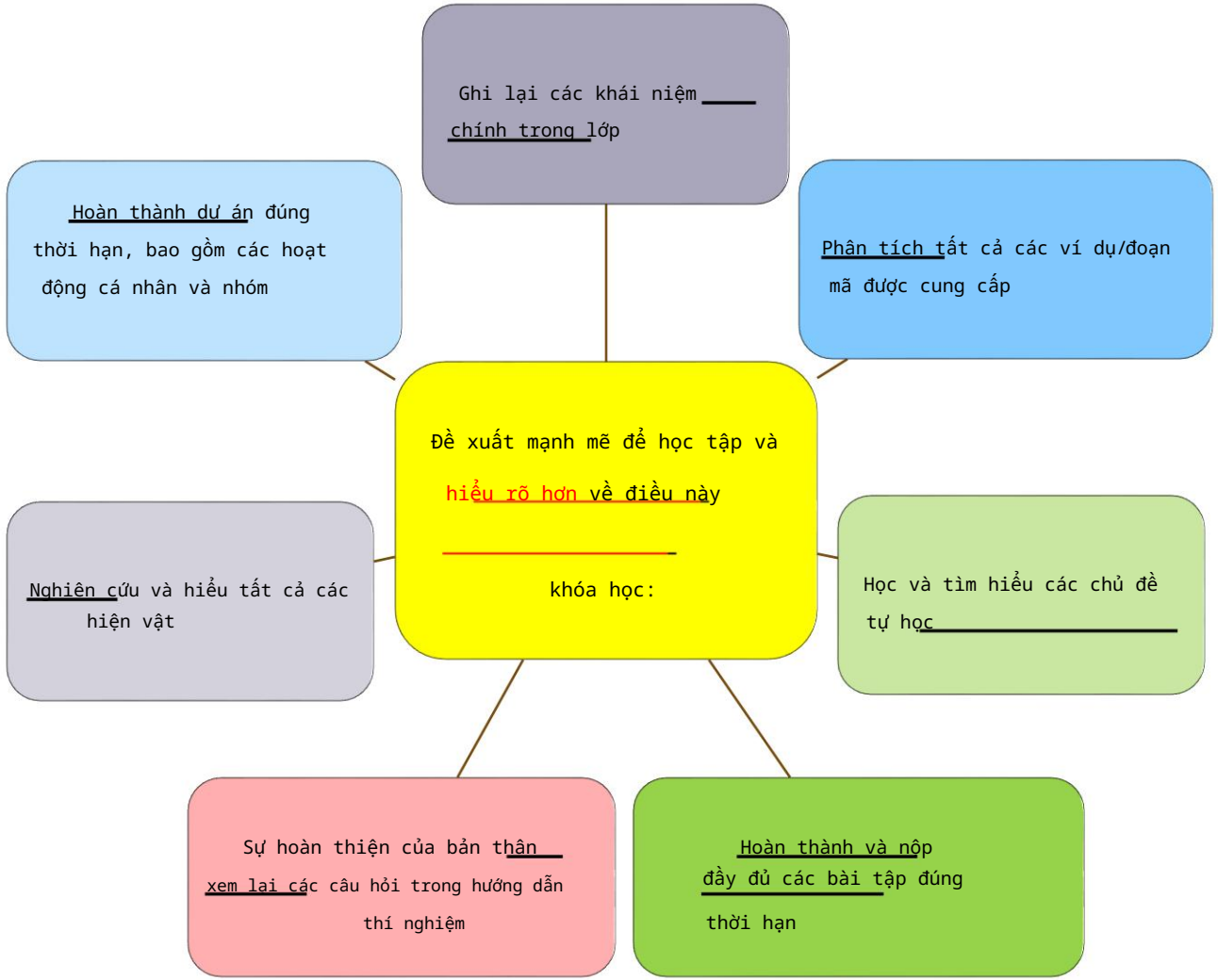
02. Trừu tượng

- Lớp trừu tượng
- Giao diện

Mục tiêu bài học



Phương pháp học tập



Phần 1

ĐA HÌNH

Tổng quan về đa hình

Đa hình, có nghĩa là "nhiều dạng", là một khái niệm quan trọng khác trong Java đi đôi với tính trừu tượng.

Nó cho phép bạn xử lý các đối tượng thuộc các loại khác nhau theo cùng một cách mà không cần biết chi tiết chính xác của chúng. Điều này làm cho mã của bạn linh hoạt và mạnh mẽ hơn.

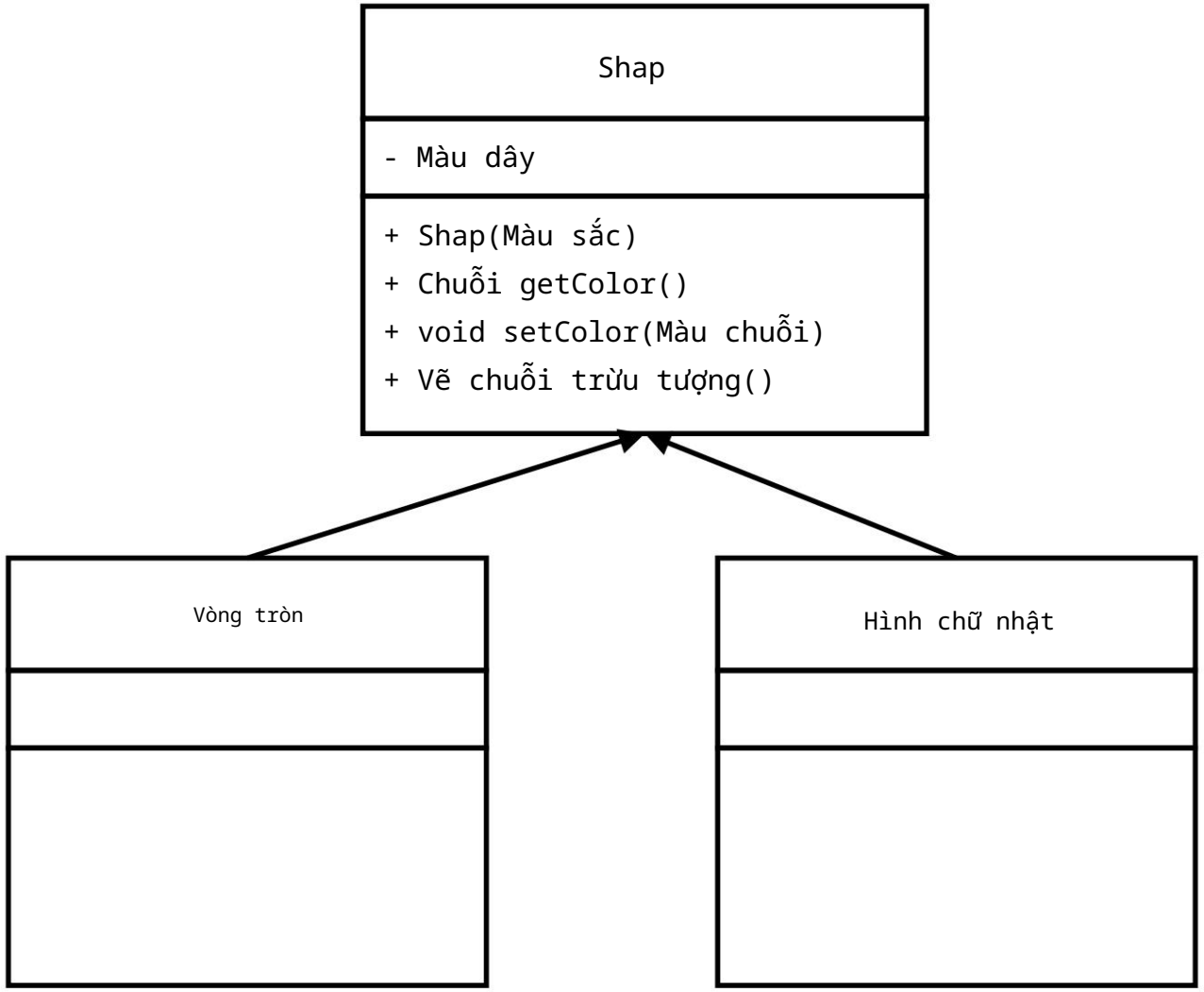
Có hai loại đa hình trong java:

Đa hình thời gian biên dịch: nạp chồng phương thức .

Đa hình thời gian chạy: ghi đè phương thức .

một cái tên, nhiều hình thức.

Ví dụ đa hình



Ví dụ đa hình

```
lớp công khai Hình dạng {
      màu chuỗi riêng tư ;

      Hình dạng công khai ( Màu chuỗi) {
            this.setColor(color);
      }

      Chuỗi công khai getColor() {
            trả lại màu sắc;
      }

      public void setColor(String color) {
            this.color = màu;
      }

      rút thăm khoảng trống công khai () {
        trở lại "Tôi là một " + cái này.màu + " hình dạng.";
      }
}
```

```
lớp công khai Vòng tròn mở rộng Hình dạng {
      Vòng kết nối công khai ( Màu chuỗi) {
            siêu (màu);
      }
      @Ghi đè
      Vẽ chuỗi công khai() {
            trở lại "Tôi là một " + this.getColor() + " vòng tròn.";
      }
}
```

```
lớp công khai Hình chữ nhật mở rộng Hình dạng {
      Hình chữ nhật công khai ( Màu chuỗi) {
            siêu (màu);
      }

      @Ghi đè
      Vẽ chuỗi công khai() {
            trở lại "Tôi là một " + this.getColor() + " hình chữ nhật.";
      }
}
```

Ví dụ đa hình

```
public class PolymorphismExample { Private
      List<Shape> Shapes = new ArrayList<Shape>();

      Ví dụ đa hình công khai () {

            Hình dạng hình dạng = Hình dạng mới
            ("Vàng"); Hình dạng myFirstCircle = Vòng tròn
            mới ("Đỏ"); Vòng tròn mySecondCircle = Vòng tròn mới
            ("Xanh"); Hình chữ nhật myFirstRectangle = Hình chữ nhật mới
            ("Xanh"); hình
            dạng.add (hình dạng); hình
            dạng.add (myFirstCircle); hình
            dạng.add (mySecondCircle); hình dạng.add(myFirstRectangle);


      } public List<Shape> getShapes() { trả
             về hình dạng;
      }

      public static void main(String[] args) {
            Ví dụ về Đa hình Ví dụ = Ví dụ đa hình mới (); for (Hình dạng :
            example.getShapes()) { System.out.println(shape.draw());


            }
      }
}
```

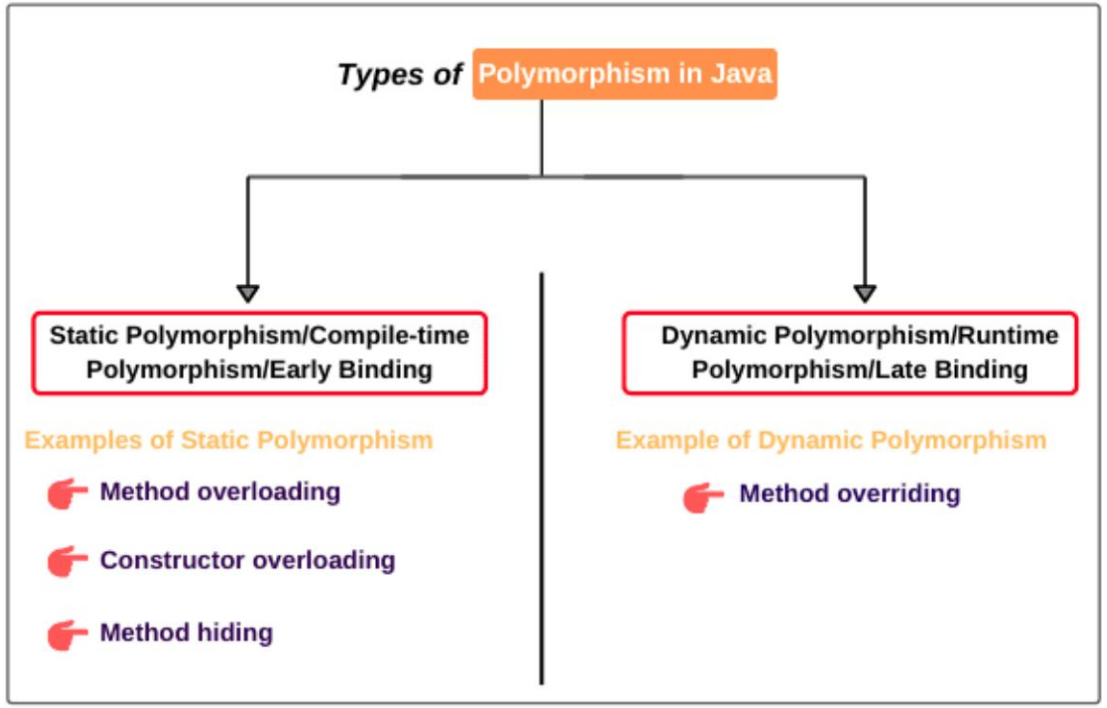
```
Đầu ra: Tôi
      là người có hình dạng Màu vàng.
      Tôi là vòng tròn Đỏ.
      Tôi là vòng tròn màu xanh.

      Tôi là một hình chữ nhật màu xanh lá cây.
```

Các loại đa hình

Có hai loại đa hình trong Java:

- Đa hình tĩnh còn được gọi là đa hình thời gian biên dịch
- Đa hình động còn được gọi là đa hình thời gian chạy



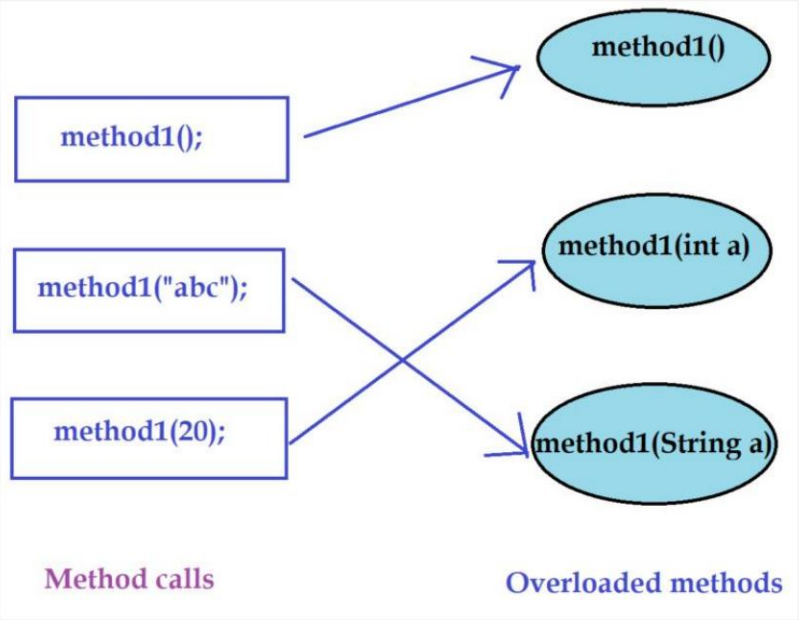
Các loại đa hình

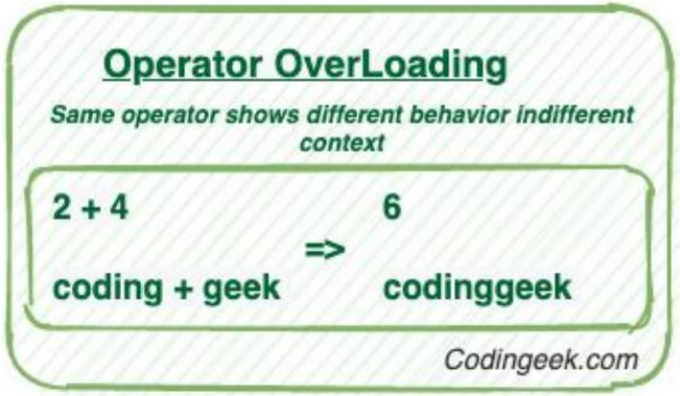
Đa hình thời gian biên dịch (hoặc đa hình tĩnh)

Đa hình được giải quyết trong thời gian biên dịch được gọi là đa hình tĩnh.

Trình biên dịch Java liên kết các lệnh gọi phương thức với định nghĩa/nội dung phương thức trong quá trình biên dịch. Vì vậy, điều này loại đa hình còn được gọi là đa hình thời gian biên dịch trong Java.

Nạp chồng phương thức và nạp chồng toán tử là những ví dụ về đa hình thời gian biên dịch





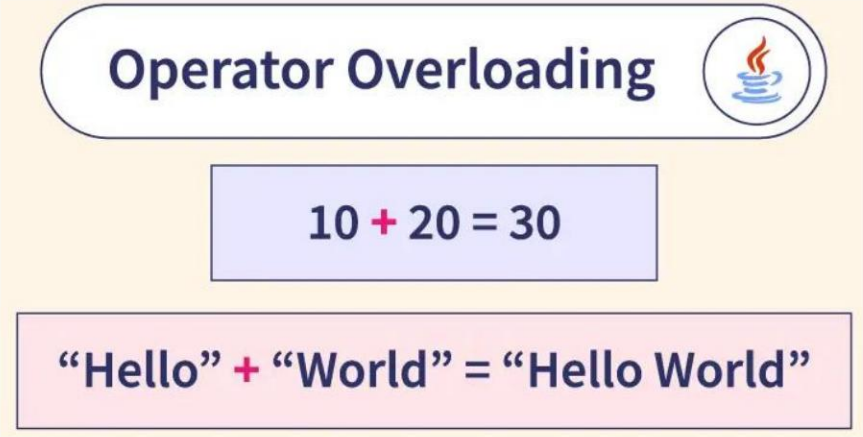
Quá tải toán tử

Quá tải toán tử là quá trình nạp chồng toán tử để thực hiện các chức năng khác nhau.

Quá tải toán tử đề cập đến khả năng xác định lại ý nghĩa của các toán tử hiện có (như +, -, *, v.v.) cho các kiểu dữ liệu tùy chỉnh, tùy thuộc vào các toán hạng liên quan.

Trên thực tế, Java không hỗ trợ nạp chồng toán tử theo nghĩa cổ điển: chỉ hỗ trợ toán tử cộng được phép có các chức năng khác nhau.

Một số toán tử, như "+" để nối chuỗi, đã có sẵn hành vi tích hợp cho các kiểu dữ liệu cụ thể.



Quá tải toán tử

Tuy nhiên, Java có cách tiếp cận khác để đạt được chức năng tương tự thông qua. Vì ví dụ:

```
lớp công khai OperatorOverload {
      public void plusOperator(int num1, int num2) {
            System.out.println(" Toán tử cộng có thể cộng hai số nguyên!"                  + (num1 + num2));
      }

      public void plusOperator(String str1, String str2) {
            System.out.println(" Toán tử cộng cũng có thể nối hai chuỗi!" + (str1 + str2));
      }

      public static void main(String[] args) { Toán tử
            OperatorOverload = Toán tử mớiOverload (); Chuỗi str1 = "Dữ liệu",
            str2 = "Sự tinh tế"; int num1 = 10, num2 = 14;
            operator.plusOperator(str1,
            str2); operator.plusOperator(num1, num2);

      }
}
```

Đầu ra:

```
Toán tử cộng cũng có thể nối hai chuỗi! DataFlair Toán tử cộng có thể cộng hai
số nguyên! 24
```

Quá tải phương thức

Các phương thức được nạp chồng cho phép bạn sử dụng lại cùng một tên phương thức trong một lớp, nhưng với các đối số khác nhau (và tùy chọn, một kiểu trả về khác).

Có hai cách để nạp chồng phương thức trong java:

Bằng cách thay đổi số lượng đối số

Bằng cách thay đổi kiểu dữ liệu

Ví dụ:

```
void func() { ... }

void func(int a) { ... }

float func(double a) { ... }

float func(int a, float b) { ... }
```

add(int a, int b) và add(double x, double y) để thêm các loại dữ liệu khác nhau.

print(String message) và print(Object obj) để in các loại dữ liệu khác nhau.

Sort(int[] arr) và Sort(List<String> list) để sắp xếp các cấu trúc dữ liệu khác nhau.

Quá tải phương thức

Ví dụ:

```
lớp công khai Hình dạng
      { khu vực trống công khai
            () { System.out.println("Tìm khu vực ");
      }

      khu vực trống công khai (int r)
            { System.out.println(" Khu vực vòng tròn =     " + 3,14 *    r * r);
      }

      khu vực trống công khai (double b, double h)
            { System.out.println(" Khu vực tam giác=" + 0,5 * b * h);
      }

      khu vực trống công khai (int l, int b)
            { System.out.println(" Khu vực hình chữ nhật=" + l * b);
      }
}
```

```
public class Main
      { public static void main(String[] args) { // Tạo
            một đối tượng Shapes Shapes
            myShape = new Shapes(); myShape.area();
            myShape.area(5);
            myShape.area(6.0,
            1.2); myShape.area(6, 2);

      }
}
```

Đầu ra:

```
Tìm diện
tích Diện tích hình tròn
= 78,5 Diện tích tam giác=3,5999999999999996
Diện tích hình chữ nhật=12
```

Quá tải phương thức

Một số quy tắc chính để nạp chồng một phương thức:

Các phương thức nạp chồng phải thay đổi danh sách đối số.

Các phương thức nạp chồng có thể thay đổi kiểu trả về.

Các phương thức quá tải có thể thay đổi công cụ sửa đổi truy cập.

Một phương thức có thể được nạp chồng trong cùng một lớp hoặc trong một lớp con.

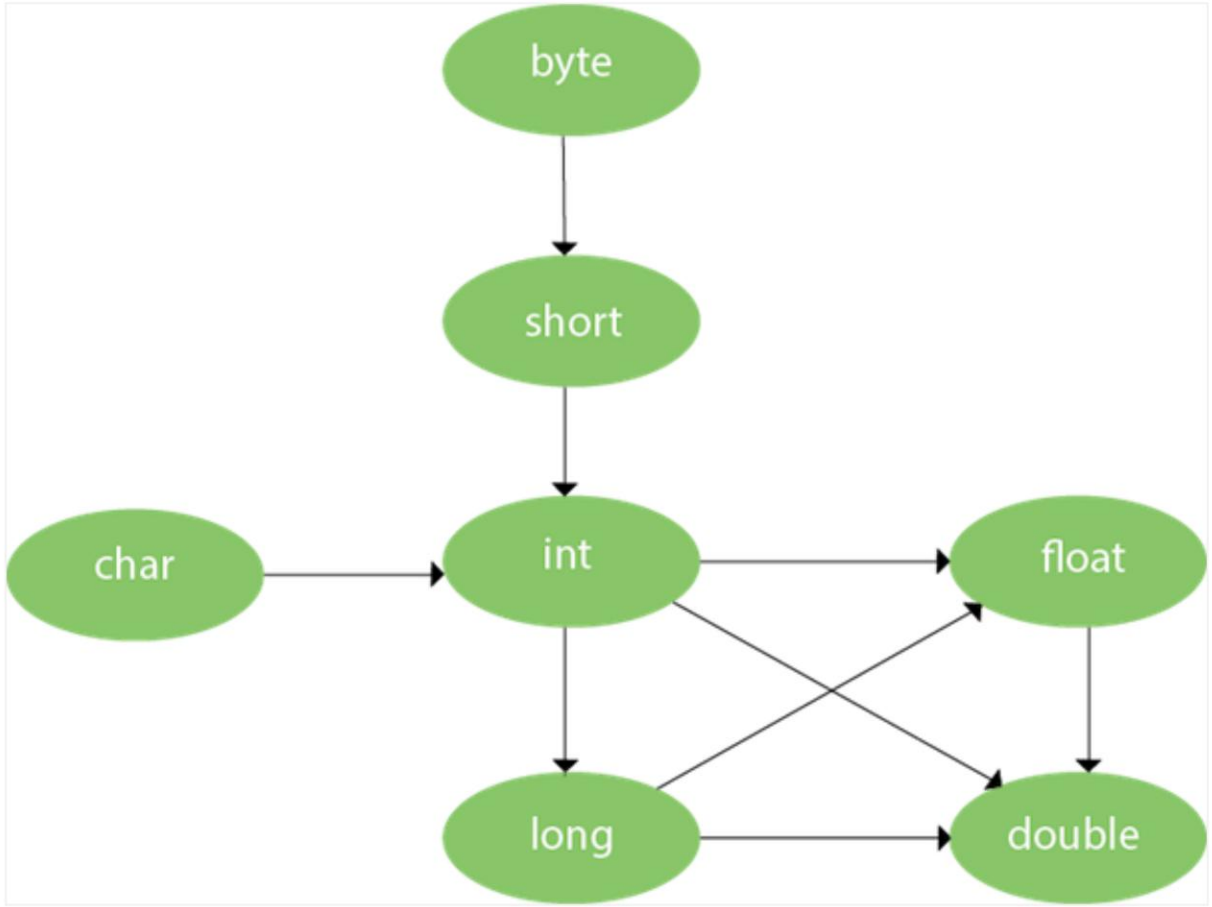
Trong một lớp con,

bạn có thể nạp chồng các phương thức được kế thừa từ siêu lớp. Các phương thức nạp chồng như vậy không ẩn hay ghi đè các phương thức siêu lớp, chúng là các phương thức mới các phương thức, duy nhất cho lớp con.

Lưu ý: Khi tôi nói chữ ký phương thức , tôi không nói về kiểu trả về của phương thức, ví dụ: nếu hai phương thức có cùng tên, cùng tham số và có kiểu trả về khác nhau thì đây không phải là ví dụ nạp chồng phương thức hợp lệ. Điều này sẽ gây ra lỗi biên dịch.

Nạp chồng phương thức và khuyến mãi kiểu

Một kiểu được thăng cấp hoàn toàn thành một kiểu khác nếu không tìm thấy kiểu dữ liệu phù hợp:



Thời gian luyện tập

Một chương trình tính toán và hiển thị số tiền thưởng để trả cho nhiều loại nhân viên khác nhau. Có 3 phòng riêng biệt được đánh số 1, 2 và 3.

Nhân viên Phòng 1 được trả thưởng dựa trên doanh số bán hàng của họ: Nếu số tiền bán hàng của họ trên 5000 USD, họ sẽ nhận được 5% doanh số bán hàng đó, nếu không họ sẽ không nhận được gì.

Nhân viên Phòng 2 được trả tiền thưởng dựa trên số lượng sản phẩm họ bán được: Họ nhận được 100 USD cho mỗi sản phẩm bán được và thêm 50 USD cho mỗi sản phẩm nếu bán được trên 25 sản phẩm; nếu họ không bán được đơn vị nào, họ sẽ không nhận được gì.

Nhân viên bộ phận 3 lắp ráp các bộ phận trong nhà máy và được thưởng 10 cent/bộ phận nếu đạt đến mức nhất định: Nhân viên bán thời gian phải lắp ráp hơn 250 bộ phận để được thưởng 10 cent/bộ phận và nhận đầy đủ -thời gian nhân viên phải tập hợp hơn 700 người.

Viết một bộ gồm 3 phương thức nạp chồng tên là getBonus() hoạt động với chương trình bên dưới, theo các thông số kỹ thuật được mô tả ở trên.

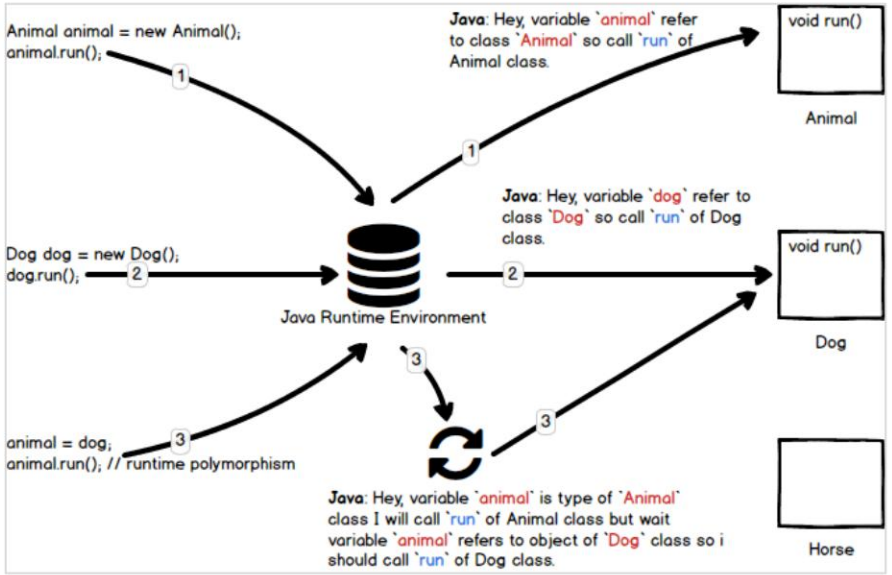
Các loại đa hình

Đa hình thời gian chạy (hoặc đa hình động):

Trong đa hình động, hành vi của một phương thức được quyết định trong thời gian chạy. JVM (Máy ảo Java) liên kết lệnh gọi phương thức với định nghĩa/nội dung phương thức trong thời gian chạy và gọi phương thức liên quan trong thời gian chạy khi phương thức đó được gọi.

- Trình biên dịch Java không biết phương thức được gọi trên một cá thể trong quá trình biên dịch.

Tính đa hình động hoặc thời gian chạy có thể đạt được/triển khai trong Java bằng Method Ghi đè.



Ghi đè phương thức

Khai báo một phương thức ở lớp con mà đã có ở lớp cha là đã biết như ghi đè phương thức.

Ưu điểm chính của việc ghi đè phương thức là:

Lớp có thể đưa ra cách triển khai cụ thể của riêng mình cho một phương thức được kế thừa mà không cần sửa đổi lớp cha (lớp cơ sở).

Quy tắc ghi đè phương thức trong Java:

Tên phương thức: phương thức phải có cùng tên với tên của lớp cha.

Danh sách đối số: phải giống với danh sách đối số của lớp cha,

Access Modifier: không thể hạn chế hơn phương thức được ghi đè của lớp cha.

Ví dụ về ghi đè phương thức/đặt giá thầu động

```
lớp Động vật có vú {
      Chuỗi makeNoise()
            { return "tiếng ồn chung";
      }

} lớp Ngựa vằn mở rộng Động vật có vú {
     @Ghi đè
      Chuỗi makeNoise()
            { return "bray";
      }

} public class ZooKeeper
      { public static void main(String[] args) { Động
            vật có vú m = new Zebra();
            System.out.println(m.makeNoise());
      }
}
```

Khi bạn gọi một phương thức thông qua một biến tham chiếu tham chiếu đến loại lớp cha, nhưng đối tượng thực tế trong thời gian chạy lại thuộc về một lớp con, liên kết động sẽ được kích hoạt.

Tra cứu thời gian chạy: Java tra cứu cách triển khai cụ thể nhất của phương thức có sẵn trong hệ thống phân cấp kế thừa của loại đối tượng thực tế.

Ghi đè "gần nhất" này được ưu tiên và được gọi khi chạy.

Đầu ra:

```
tiếng ồn ào
```

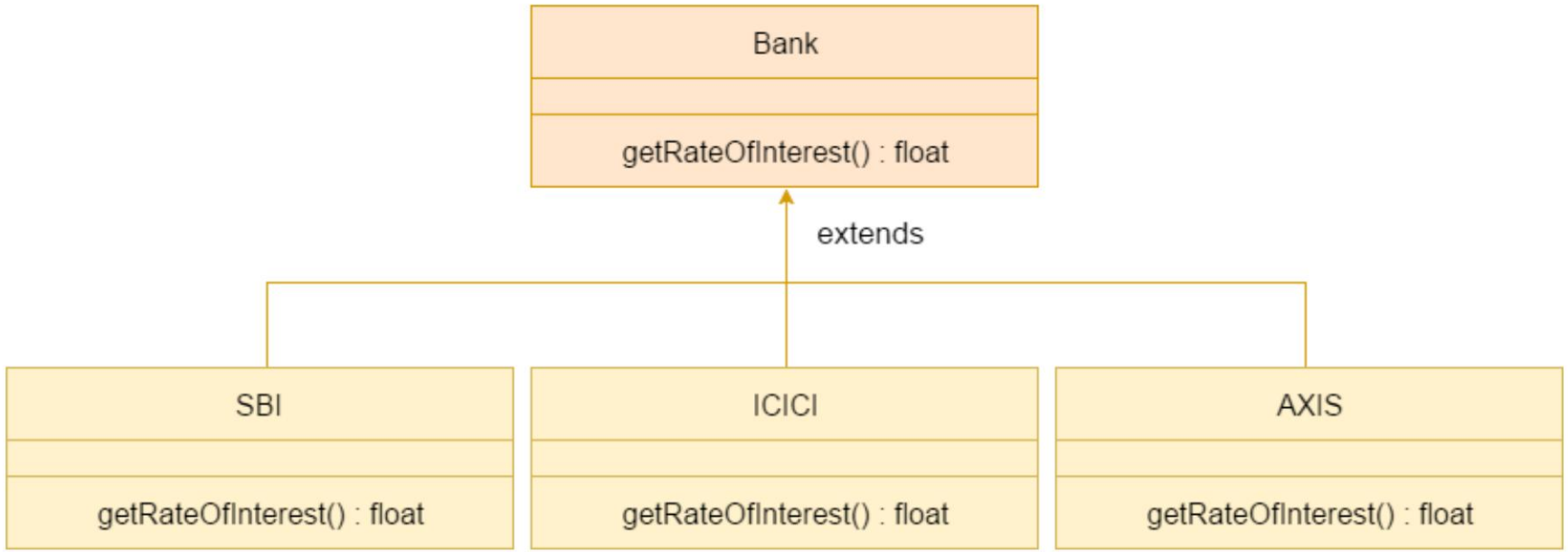
Một ví dụ thực tế về Ghi đè phương thức

Hãy xem xét một kịch bản trong đó Bank là một lớp cung cấp chức năng để nhận được lãi suất.

Tuy nhiên, lãi suất sẽ khác nhau tùy theo ngân hàng.

Ví dụ: các ngân hàng SBI, ICICI và AXIS có thể cung cấp 8%, 7% và 9%

tỷ lệ lãi suất.



Một ví dụ thực tế về Ghi đè phương thức

```
lớp Ngân hàng {
   float getRateOfInterest() {
     trả về 0;
   }
}

lớp SBI mở rộng Ngân hàng {
   float getRateOfInterest() {
     trả về 8,4f;
   }
}
lớp ICICI mở rộng Ngân hàng {
   float getRateOfInterest() {
     trả về 7,3f;
   }
}
lớp AXIS mở rộng Ngân hàng {
   float getRateOfInterest() {
     trả về 9,7f;
   }
}
```

```
lớp công khai TestPolymorphism {
      public static void main(String args[]) {
         Ngân hàng b;
         b = SBI mới ();
         System.out.println(" Lãi suất SBI: " +
                                          b.getRateOfInterest());

         b = ICICI mới ();
         System.out.println("ICICI Lãi suất: " +
                                           b.getRateOfInterest());

         b = AXIS mới ();
         System.out.println(" Tỷ lệ lãi suất AXIS: " +
                                            b.getRateOfInterest());
      }
}
```

Đầu ra:

```
Lãi suất SBI: 8,4

Lãi suất ICICI: 7,3

Lãi suất AXIS: 9,7
```

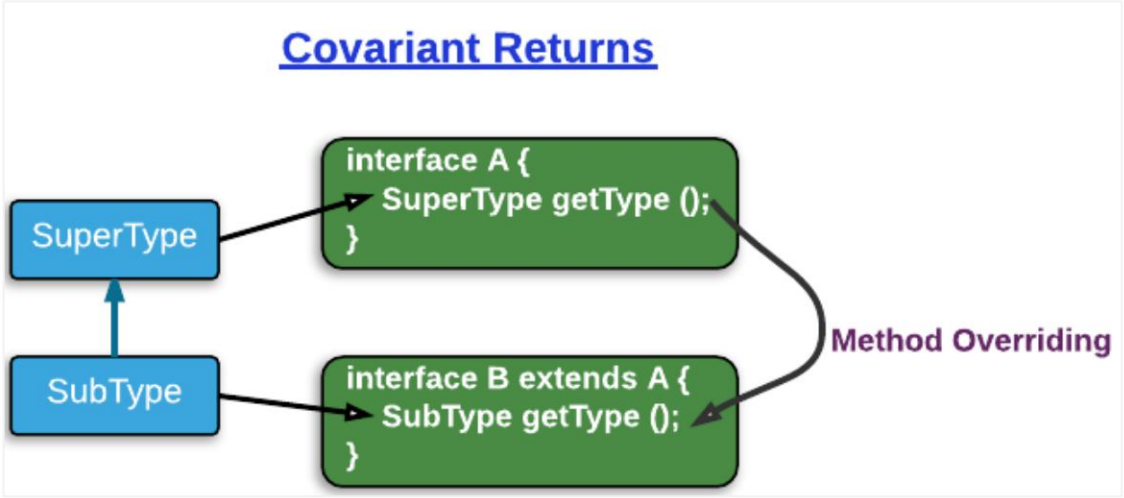
Ghi đè phương thức: Kiểu trả về hiệp biến

Các kiểu trả về hiệp biến trong ghi đè phương thức cho phép một lớp con trả về một kiểu con của kiểu trả về được khai báo trong phương thức lớp cha.

Điều này mang lại tính linh hoạt cao hơn và an toàn hơn khi mở rộng chức năng kế thừa hệ thống phân cấp.

Khi một lớp con ghi đè một phương thức từ lớp cha của nó, nó có thể: Có cùng kiểu trả về với phương thức cha (ghi đè truyền thống). Có một kiểu con của kiểu trả về của phương thức cha (ghi đè hiệp biến).

Điều này cho phép lớp con trả về một phiên bản cụ thể hơn của dữ liệu được lớp cha trả về phương pháp.



Bóng tối

Cái này được gọi là Shadowing—tên trong lớp Dog bóng tên trong lớp Animal

```
lớp công khai Động vật {
      Tên chuỗi = "Động vật";
        tiếng nói công khai () {
             System.out.println(" nói chung!");
         }
      public static void main(String args[]) { Animal
               Animal1 = new Animal(); Động vật Animal2
               = Chó mới (); Chó chó = Chó mới ();
               System.out.println(animal1.name
               +                                                    + động vật2.name +" "       + dog.name);
      }

} lớp Chó mở rộng Động vật {
      Tên chuỗi = "Con chó";
        tiếng nói công khai () {
             System.out.println("chó nói!");
         }
}
```

Đầu ra:

```
Động vật chó động vật
```

Liên kết tĩnh và động

Liên kết tĩnh

Liên kết có thể được giải quyết tại thời điểm biên dịch bởi trình biên dịch được gọi là liên kết tĩnh hoặc liên kết sớm.

Sự ràng buộc của các phương thức tĩnh, riêng tư và cuối cùng là thời gian biên dịch. Lý do là các phương thức này không thể bị ghi đè và loại của lớp được xác định tại thời điểm biên dịch.

Liên kết động

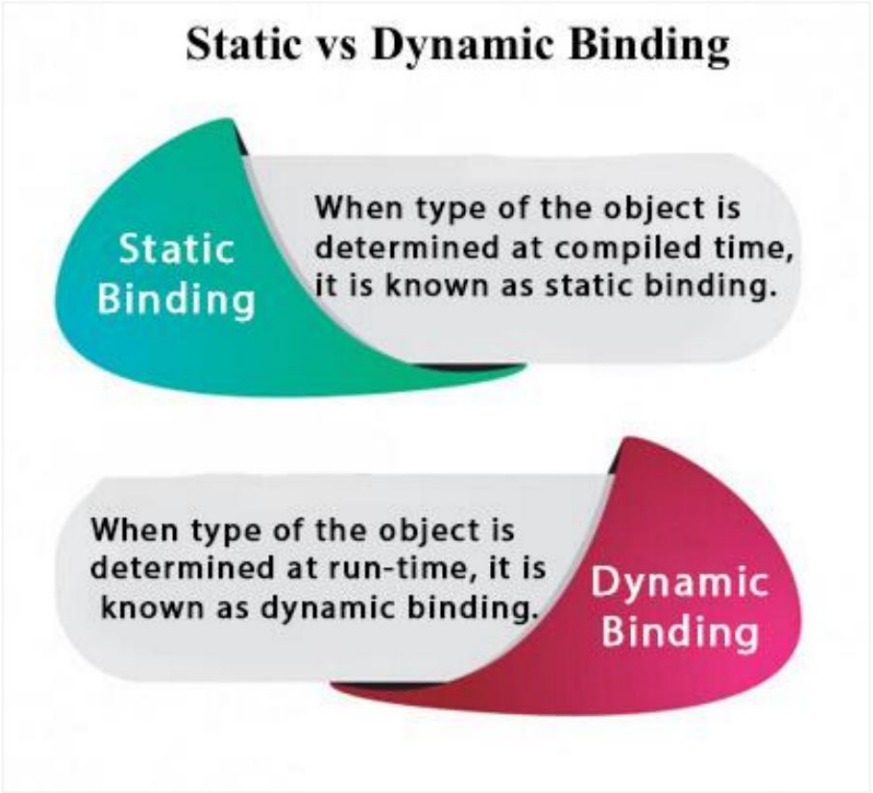
Khi trình biên dịch không thể giải quyết lệnh gọi/ liên kết tại thời điểm biên dịch, liên kết đó được gọi là Liên kết động hoặc Liên kết muộn.

Ghi đè phương thức là một ví dụ hoàn hảo về liên kết động vì việc ghi đè cả lớp cha và lớp con có cùng một phương thức và trong trường hợp này, loại đối tượng sẽ xác định phương thức nào sẽ được thực thi.

Loại đối tượng được xác định tại thời điểm chạy nên đây được gọi là liên kết động.

Liên kết tĩnh và động

Liên kết tĩnh	Liên kết động
Xảy ra vào thời điểm biên dịch	Xảy ra trong thời gian chạy
Đối tượng thực tế không được sử dụng	Đối tượng thực tế được sử dụng
Còn được gọi là ràng buộc sớm Còn được gọi là ràng buộc muộn	
Tốc độ cao	Tốc độ thấp
Ví dụ: - Nạp chồng phương thức	Ví dụ: - Ghi đè phương thức



Ghi đè và liên kết phương thức tĩnh

Trong khi các phương thức tĩnh trong các lớp con về mặt kỹ thuật "ẩn" các phương thức cùng tên khỏi lớp cha mẹ của họ, nó không được coi là trốn tránh thực sự theo nghĩa truyền thống.

Khi gọi một phương thức tĩnh, trình biên dịch vẫn thực hiện liên kết tĩnh dựa trên kiểu của biến tham chiếu (luôn là tên lớp). Vì vậy, không phải kiểu của đối tượng xác định phương thức được gọi mà là kiểu tĩnh của tham chiếu được sử dụng.

Ví dụ:

```
class ParentClass
   { public static void staticMethod()
      { System.out.println(" Phương thức tĩnh của lớp cha");
   }
}

lớp ChildClass mở rộng ParentClass
   { public static void staticMethod()
      { System.out.println(" Phương thức tĩnh của lớp con");
   }
}
```

Ghi đè phương thức tĩnh: Liên kết tĩnh

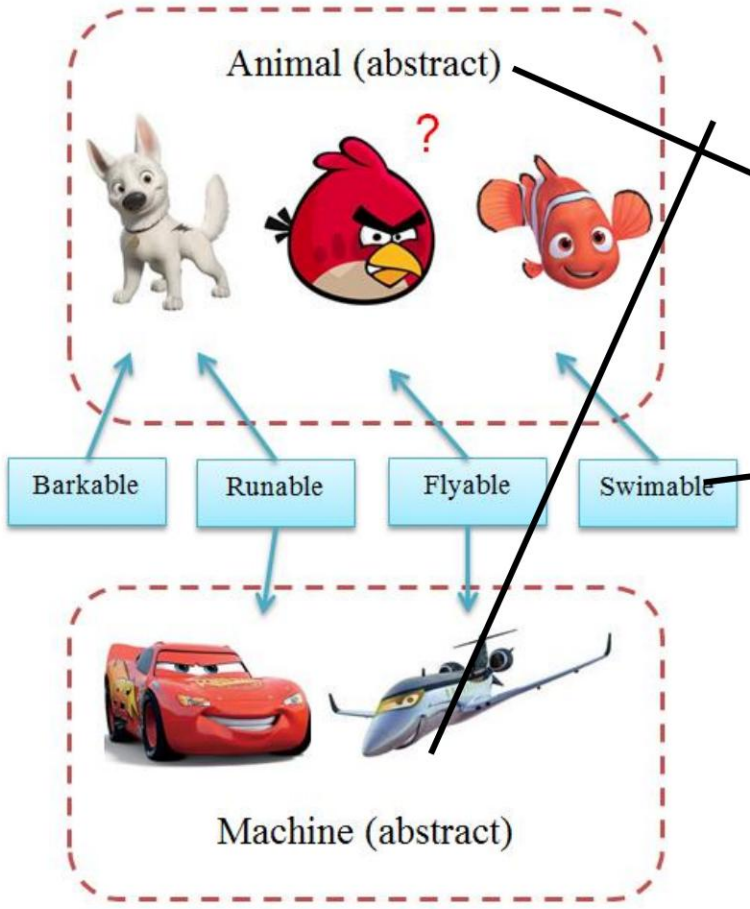
```
lớp công khai Main2 {
    public static void main(String[] args)
        { ParentClass.staticMethod(); // In "Phương thức tĩnh của lớp cha"
        ChildClass.staticMethod(); // In "Phương thức tĩnh của lớp con"
    }
}
```

Phần 2

TÓM TẮT

Tổng quan trừu tượng

Trừu tượng hóa là một quá trình ẩn các chi tiết triển khai và chỉ hiển thị chức năng cho người dùng.



Ví dụ:

Gửi sms, bạn chỉ cần gõ nội dung và gửi tin nhắn. Bạn không biết xử lý nội bộ về việc gửi tin nhắn

Nó chỉ hiển thị những điều quan trọng cho người dùng và ẩn các chi tiết bên trong

Tập trung vào những gì đối tượng làm thay vì nó làm như thế nào.

Có hai cách để đạt được tính trừu tượng trong java: Lớp trừu tượng (0 đến 100%) Giao diện (100%)

Mục đích của sự trừu tượng

Ẩn chi tiết triển khai, tập trung vào chức năng của lớp thay vì nó thực hiện như thế nào (hoạt động nội bộ).

Làm cho mã dễ hiểu và dễ sử dụng hơn, giảm tải nhận thức cho nhà phát triển.

Những thay đổi trong triển khai nội bộ có thể được thực hiện mà không ảnh hưởng đến hành vi bên ngoài, đảm bảo tính ổn định và giảm nguy cơ phá vỡ mã hiện có.

Giúp sửa lỗi và thêm các tính năng mới dễ dàng hơn vì các sửa đổi được bản địa hóa bên trong lớp trừu tượng.

Các lớp con có thể kế thừa và mở rộng chức năng hiện có, tiết kiệm thời gian phát triển và cố gắng.

Các lớp và giao diện trừu tượng xác định các hợp đồng cho hành vi mà không chỉ rõ chi tiết thực hiện.

Lớp trừu tượng

Lớp được khai báo là lớp trừu tượng thì được gọi là lớp trừu tượng.

Nó có thể có các phương thức trừu tượng và không trừu tượng.

Nó cần được mở rộng và thực hiện phương pháp của nó.

Không thể khởi tạo được.

Các quy tắc cho lớp trừu tượng Java:

1. • Lớp trừu tượng phải được khai báo bằng từ khóa trừu tượng.
2. • Nó có thể có các phương thức trừu tượng và không trừu tượng.
3. • Nó không thể được khởi tạo.
4. • Nó có thể có hàm tạo và cả các phương thức tĩnh nữa.
5. • Nó có thể có các phương thức cuối cùng sẽ buộc lớp con không thay đổi nội dung của phương thức.

Lớp trừu tượng và phương thức trừu tượng

Một lớp trừu tượng không thể được khởi tạo (bạn không được phép tạo đối tượng của lớp Trừu tượng), nhưng chúng có thể được phân lớp.

```
Động vật thú = Mèo mới ();
```

Một lớp con thường cung cấp các triển khai cho tất cả các phương thức trừu tượng trong lớp cha của nó.

Tuy nhiên, nếu không thì lớp con cũng phải được khai báo là trừu tượng.

Các phương pháp trừu tượng:

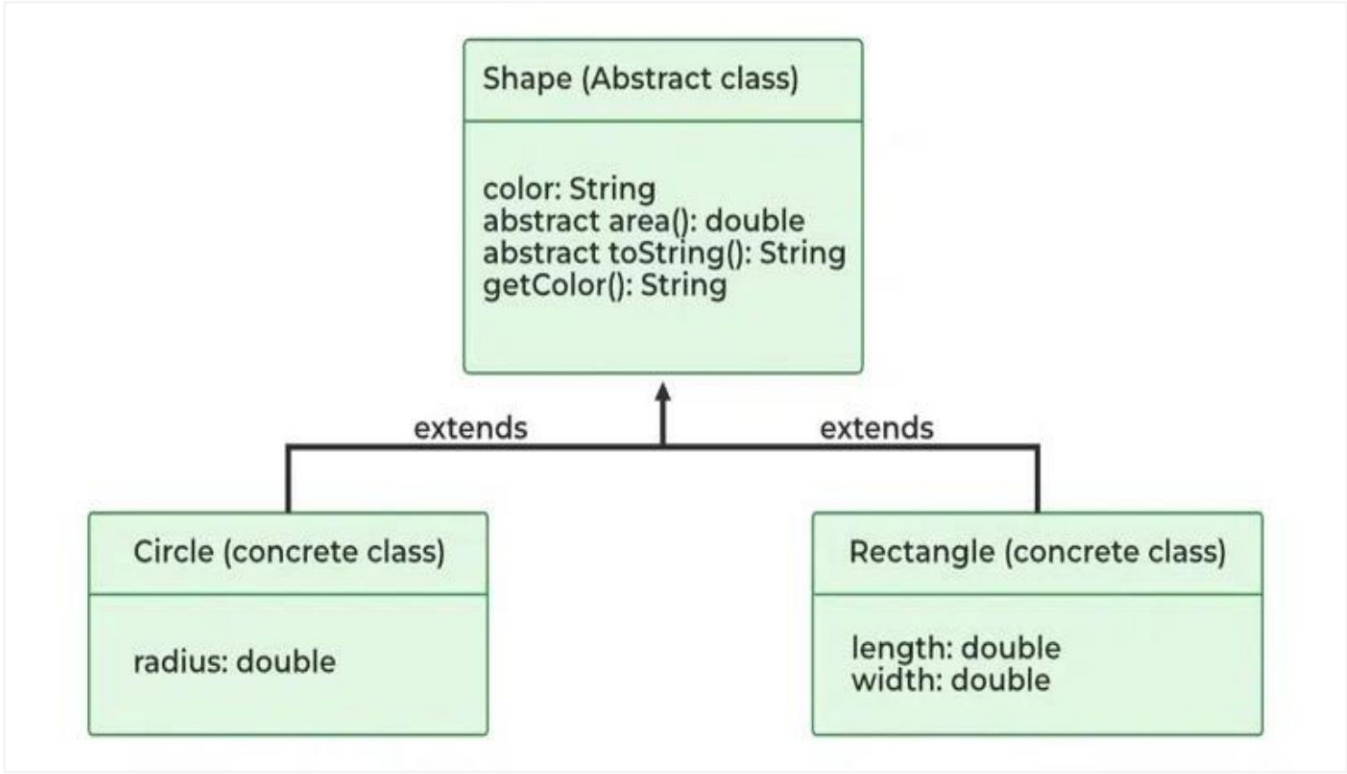
Phương thức trừu tượng là phương thức được khai báo mà không thực hiện. Ví dụ:

```
trừu tượng void moveTo(int x, int y);
```

Nếu một lớp bao gồm các phương thức trừu tượng thì bản thân lớp đó phải được khai báo là trừu tượng.

Ví dụ trừu tượng

Ví dụ:



Ví dụ trừu tượng

Ví dụ:

```
lớp trừu tượng công khai Hình dạng {
   Màu dây ;

   // đây là các phương thức trừu
   tượng vùng đôi trừu tượng ();

   // lớp trừu tượng có thể có hàm tạo public Shape(String color)
   { System.out.println(" Hàm tạo hình dạng
     được gọi"); this.color = màu; }



   // đây là một phương thức cụ thể public
   String getColor() {
      trả lại màu sắc;
   }
}
```

```
lớp Vòng tròn mở rộng Hình dạng {
  bán kính gấp đôi ;

  public Circle(String color, doublebán kính)
    { // gọi hàm tạo Shape
    super(color);
    System.out.println(" Xây dựng vòng tròn được gọi");
    this.bán kính = bán
  kính; }

  @Override
  double khu vực()
     { return Math.PI * Math.pow(radius, 2);
  }

  @Ghi đè
  chuỗi công khai toString() {
     return "Màu hình tròn là"     + super.getColor() + ", khu
                                     diện tích là:          vực(); "và

  }
}
```

Ví dụ trừu tượng

Ví dụ:

```
lớp Hình chữ nhật mở rộng Hình dạng
   { chiều dài gấp
   đôi ; chiều rộng gấp đôi ;

   public Rectangle(String color, double length, double width) { // gọi Shape
      constructor super(color);

      System.out.println(" Hàm tạo hình chữ nhật được gọi"); this.length
      = chiều dài; this.width =
      chiều rộng;
   }

   @Ghi đè diện
   tích gấp đôi ()
      { chiều dài trả về * chiều rộng;
   }

   @Ghi đè
   chuỗi công khai toString() {
      return "Màu hình chữ nhật là"          + super.getColor() + " và
                                             diện tích là : + Area();

   }
}
```

```
Kiểm tra lớp công khai {

    public static void main(String[] args) {

       Hình dạng s1 = Vòng tròn mới ("Đỏ", 2.2);

       Hình dạng s2 = Hình chữ nhật mới ("Vàng", 2, 4);

       System.out.println(s1.toString());

       System.out.println(s2.toString());

    }

}
```

Đầu ra:

```
Hàm tạo hình dạng được gọi là
Hàm tạo vòng tròn được gọi

Hàm tạo hình dạng được gọi là

Hàm tạo hình chữ nhật được gọi là
Màu hình tròn là Red và diện tích là: 15.205308443374602

Màu hình chữ nhật là màu vàng và diện tích là: 8,0
```

Giao diện

Giao diện trong Java là bản thiết kế của một lớp. Nó có các hằng số tĩnh và các phương thức trừu tượng.

Giao diện trong Java là một cơ chế để đạt được sự trừu tượng hóa.

Chỉ có thể có các phương thức trừu tượng trong giao diện Java chứ không phải phần thân phương thức.

Nó được sử dụng để đạt được tính trừu tượng và đa kế thừa trong Java.

Tại sao nên sử dụng giao diện Java?

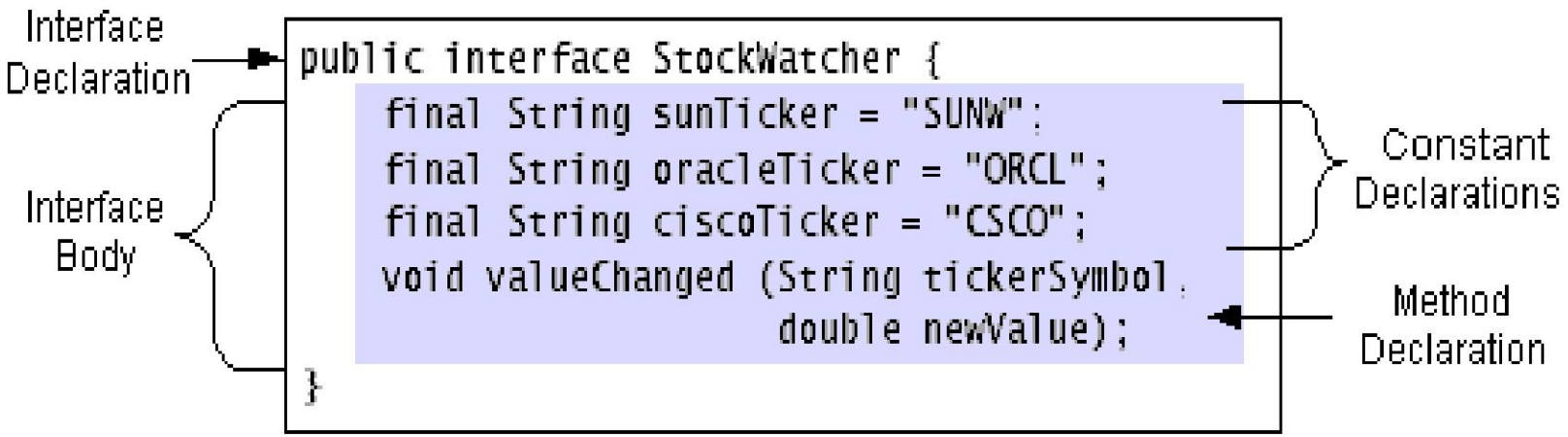
1. • Nó được sử dụng để đạt được sự trừu tượng.
2. • Bằng giao diện, chúng tôi có thể hỗ trợ chức năng đa kế thừa.
3. • Nó có thể được sử dụng để đạt được khớp nối lỏng lẻo.

Giao diện

Cú pháp:

```
giao diện [công khai] <InterfaceName>[mở rộng SuperInterface] {
        // Thân giao diện
   }
```

Ví dụ:

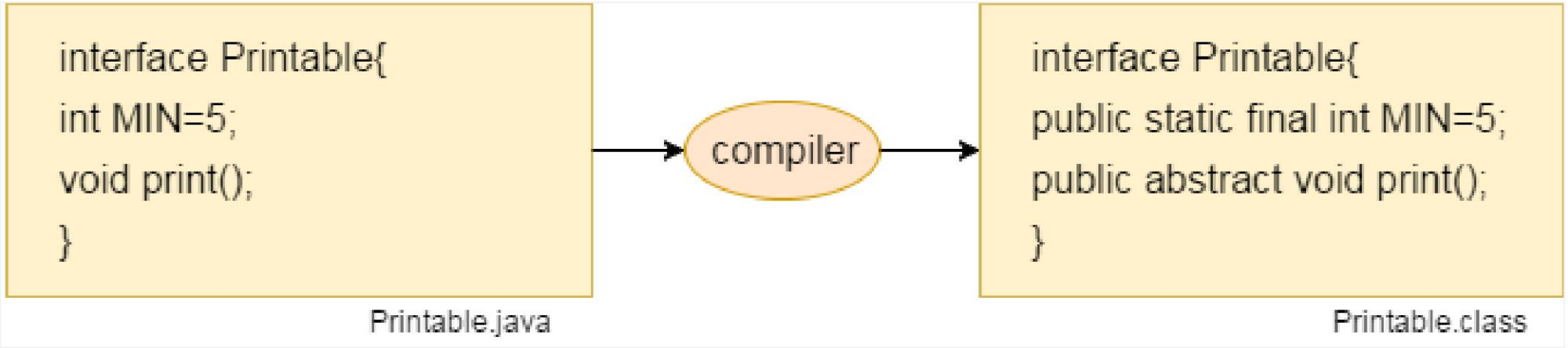


Giao diện

Trình biên dịch Java thêm các từ khóa công khai và trừu tượng trước phương thức giao diện.

Hơn nữa, nó thêm các từ khóa công khai, tĩnh và cuối cùng trước các thành viên dữ liệu.

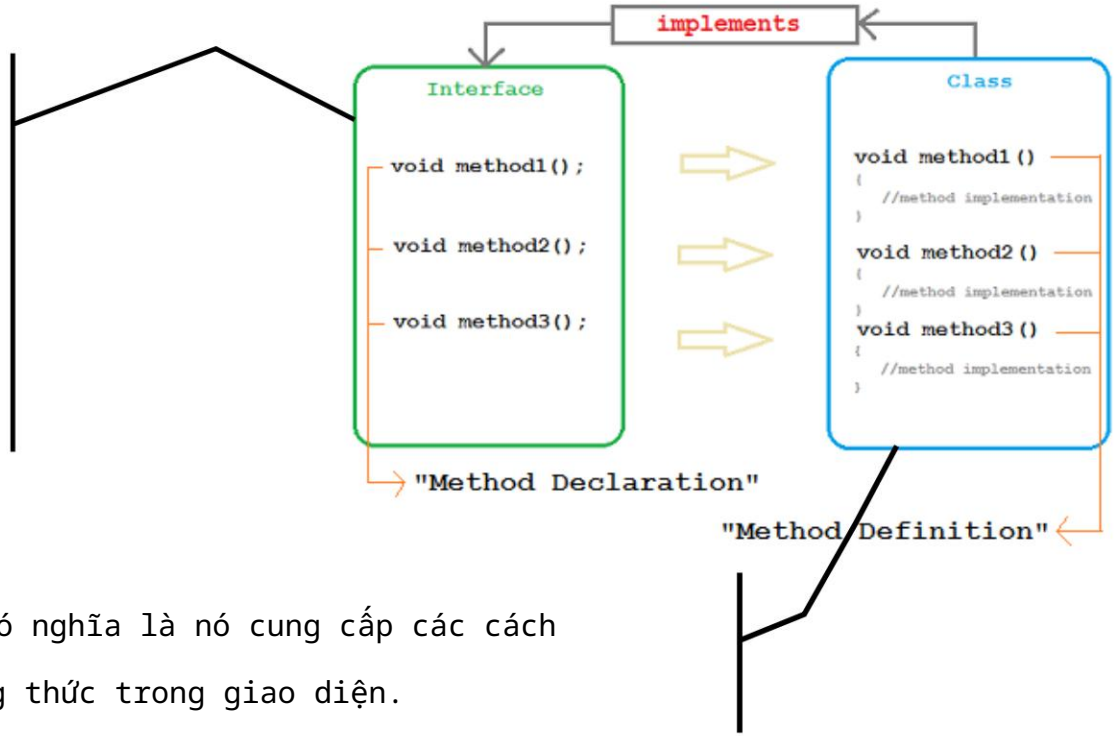
Xét hình sau:



Giao diện

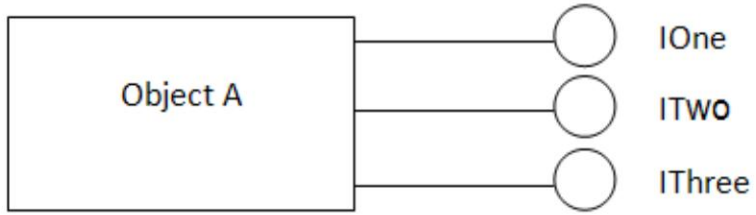
Giao diện là định nghĩa của nguyên mẫu phương thức và có thể là một số hằng số (trường cuối cùng tĩnh).

Một giao diện không bao gồm việc thực hiện bất kỳ phương pháp nào.

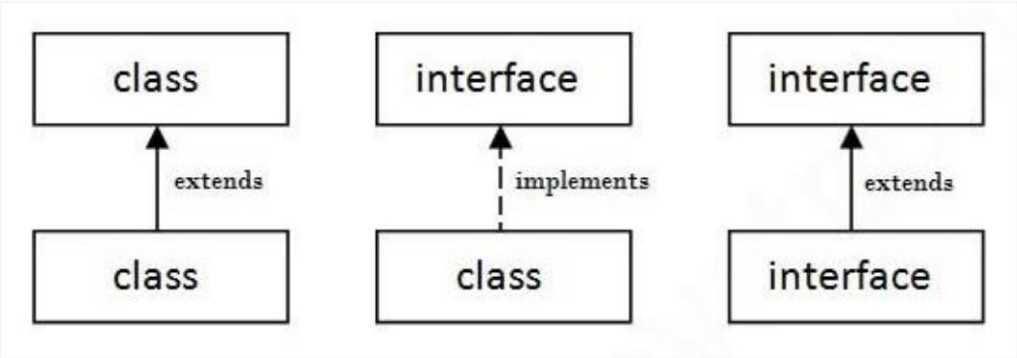


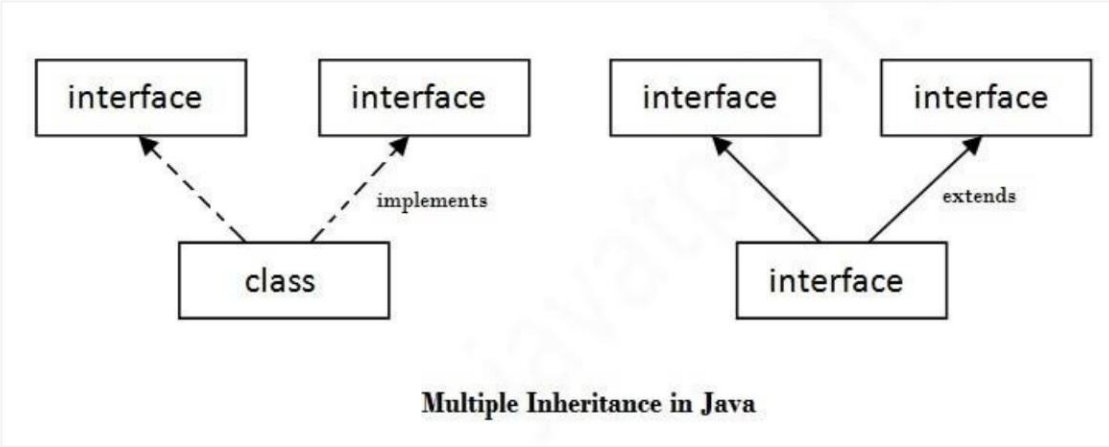
Một lớp có thể triển khai một giao diện, điều này có nghĩa là nó cung cấp các cách triển khai cho tất cả các phương thức trong giao diện.

Các lớp Java có thể triển khai bất kỳ số lượng giao diện nào (kế thừa nhiều giao diện).



Mối quan hệ giữa lớp và giao diện





Lớp học	Giao diện
Trong lớp, bạn có thể khởi tạo các biến và tạo một đối tượng.	Trong một giao diện, bạn không thể khởi tạo các biến và tạo ra một đối tượng.
Một lớp có thể chứa các phương thức cụ thể (có triển khai)	Giao diện không thể chứa các phương thức cụ thể (có triển khai)
Các chỉ định truy cập được sử dụng với các lớp là riêng tư, được bảo vệ và công khai.	Trong Giao diện chỉ có một thông số xác định được sử dụng - công khai.

Ví dụ về giao diện Java 1: Ngân hàng

Hãy xem một ví dụ khác về giao diện java cung cấp việc triển khai Bank giao diện.

```
giao diện Ngân
  hàng { float
rateOfInterest(); }

lớp SBI triển khai Ngân hàng {
   tỷ lệ thả nổi công khaiOfInterest ()
      { return 9.15f;
   }
}

lớp PNB triển khai Ngân hàng {
   tỷ lệ thả nổi công khaiOfInterest ()
      { return 9.7f;
   }
}
```

```
lớp công khai TestInterface {

   public static void main(String[] args) {

      Ngân hàng b = SBI mới
      (); System.out.println("ROI: " + b.rateOfInterest());
   }
}
```

Đầu ra:

```
ROI: 9,15
```

Giao diện Ví dụ 2

Ví dụ:

```
giao diện công cộng Chuyển tiếp
    { void drive();
}

giao diện công cộng Dừng
    { void park();
}

giao diện công cộng Tốc độ
    { void turbo();
}

lớp công khai GearBox
    { public void move() {
}
```

```
lớp Tự động mở rộng GearBox
                                    thực hiện Chuyển tiếp, Dừng, Tốc độ { public
     void drive()
          { System.out.println("drive()");

     } public void park()
          { System.out.println("park()");

     } public void turbo()
          { System.out.println("turbo()");

     } public void move()
          { System.out.println("move()");
     }
}
```

Giao diện Ví dụ 2

Ví dụ:

```
public class Car
     { public static void hành trình (Chuyển tiếp x)
         { x.drive();

     } public static void park(Stop x)
         { x.park();

     } public static void race(Speed x)
         { x.turbo();

     } public static void move(GearBox x) { x.move();


     } public static void main(String[] args) {
         Tự động tự động = Tự động mới (); hành
         trình (tự động); // Giao diện Forward
         park(auto); // Giao diện Dừng cuộc
         đua(auto); // Giao diện Tốc độ di
         chuyển(tự động); // lớp GearBox
    }
}
```

Đa kế thừa trong Java theo giao diện

Đa kế thừa không được hỗ trợ trong trường hợp lớp vì sự mơ hồ. Tuy nhiên, nó được hỗ trợ trong trường hợp có giao diện vì không có sự mơ hồ. Đó là do việc triển khai nó được cung cấp bởi lớp triển khai.

Ví dụ:

```
giao diện Có thể in được
   { void print(); }


giao diện Có thể hiển thị
   { void print();
}

lớp A4 triển khai Có thể in, Hiển thị {
   public void print()
      { System.out.println("In và hiển thị tài liệu");
   }
}

lớp công khai TestInterface2 { public
static void main(String[] args) { A4 a4 = new A4();
     a4.print();

   }
}
```

Đầu ra:

```
In và hiển thị tài liệu
```

Phương thức mặc định Java 8 trong giao diện

Kể từ Java 8, chúng ta có thể có nội dung phương thức trong giao diện. Nhưng chúng ta cần đặt nó làm mặc định phương pháp.

Ví dụ:

```
giao diện Có thể vẽ được
   { void draw();

   default void msg()
      { System.out.println(" phương thức mặc định"); }


} lớp Hình chữ nhật triển khai có thể vẽ được {
   rút thăm khoảng trống công khai () {
      System.out.println("vẽ hình chữ nhật"); }

}

lớp công khai TestInterfaceDefault { public
       static void main(String[] args) { Drawable d = new
            Rectangle(); d.draw(); d.msg();


       }
}
```

Đầu ra:

```
vẽ hình chữ nhật
phương pháp mặc định
```

Bạn có thể ghi đè việc triển khai mặc định bằng phiên bản của riêng mình.

Phương thức mặc định Java 8 trong giao diện

Ví dụ: Giao

diện có thể so sánh với phương thức so sánh mặc định : Cung cấp logic so sánh cơ bản mà các lớp triển khai có thể sử dụng hoặc ghi đè cho các nhu cầu cụ thể. Giao diện danh

sách với phương thức sắp xếp() mặc định : Cung cấp thuật toán sắp xếp tiêu chuẩn mà việc triển khai danh sách có thể tận dụng hoặc thay thế bằng logic sắp xếp tùy chỉnh.

```
@SuppressWarnings({"unchecked", "rawtypes"})
mặc định voidsort (Comparator<? super E> c) {
    Đối tượng[] a = this.toArray();
    Arrays.sort(a, (Bộ so sánh) c);
    ListIterator<E> i = this.listIterator(); for (Đối
    tượng e : a) { i.next();
       i.set((E)
       e);
    }
}
```

Phương thức tĩnh Java 8 trong giao diện

Kể từ Java 8, chúng ta có thể có phương thức tĩnh trong giao diện.

Ví dụ:

```
giao diện có thể vẽ được {
   làm mất hiệu lực vẽ();

   khối int tĩnh (int x) {
      trả về x x;     * x *
   }
}

lớp Hình chữ nhật thực hiện Drawable {
   rút thăm khoảng trống công khai () {
      System.out.println("vẽ hình chữ nhật");
   }
}

lớp công khai TestInterfaceStatic {
      public static void main(String[] args) {
            Có thể vẽ d = Hình chữ nhật mới ();
            d.draw();
            System.out.println(Drawable.cube(3));
      }
}
```

Đầu ra:

```
vẽ hình chữ nhật
27
```

Lớp trừu tượng và giao diện

Lớp trừu tượng và giao diện đều được sử dụng để đạt được sự trừu tượng nơi chúng ta có thể khai báo các phương pháp trừu tượng Cả lớp trừu tượng và giao diện đều không thể được khởi tạo.

Nhưng có nhiều điểm khác biệt giữa lớp trừu tượng và giao diện được đưa ra dưới đây.

STT	Lớp trừu tượng	Giao diện
1	Lớp trừu tượng có thể có các phương thức trừu tượng và không trừu tượng .	Giao diện chỉ có thể có các phương thức trừu tượng.
2	Lớp trừu tượng không hỗ trợ đa kế thừa.	Giao diện hỗ trợ đa kế thừa.
3	Lớp trừu tượng có thể có các biến cuối cùng, không phải cuối cùng, tĩnh và không tĩnh.	Giao diện chỉ có các biến tĩnh và cuối cùng.
4	Lớp trừu tượng có thể có các phương thức tĩnh, phương thức chính và hàm tạo.	Giao diện không được có phương thức tĩnh, phương thức chính hoặc hàm tạo.
5	Lớp trừu tượng có thể cung cấp việc triển khai giao diện.	Giao diện không thể cung cấp việc triển khai lớp trừu tượng.
6	Từ khóa abstract được sử dụng để khai báo lớp trừu tượng.	Từ khóa giao diện được sử dụng để khai báo giao diện.

7	Ví dụ:
	`lớp trừu tượng công khai Hình dạng {`
	`trừu tượng công khai void draw();`
	`}`
		Ví dụ:
		`giao diện công cộng có thể vẽ được {`
		`làm mất hiệu lực vẽ();`
		`}`

Bản tóm tắt

Tính đa hình, có nghĩa là "nhiều hình thức",

là khả năng xử lý một đối tượng của bất kỳ lớp con nào của lớp cơ sở như thể nó là một đối tượng của lớp cơ sở.

Lớp trừu tượng là lớp có thể chứa các phương thức trừu tượng và các phương thức được triển khai.

Một phương thức trừu tượng là một phương thức không có phần thân và được khai báo bằng từ dành riêng abstract

Giao diện là tập hợp các khai báo hằng và phương thức.

Khi một lớp triển khai một giao diện, nó phải khai báo và cung cấp phần thân phương thức cho mỗi phương thức trong giao diện

Tài liệu tham khảo

https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

https://www.mygreatlearning.com/blog/polymorphism-in-java/

https://www.javatpoint. com/runtime-polymorphism-in-java

https://www.geeksforgeeks.org/polymorphism-in-java/

Câu hỏi

CẢM ƠN!