**BÁO CÁO MÔN JAVA**

1. **Class**

Một lớp là một nhóm các đối tượng có các thuộc tính chung. Nó là một mẫu hoặc bản thiết kế mà từ đó các đối tượng được tạo ra. Nó là một thực thể logic. Nó không thể là vật chất.

Một lớp trong Java có thể chứa:

* **Fields**
* **Methods**
* **Constructors**
* **Blocks**
* **Nested class and interface**

Cú pháp khai báo một lớp:

1. **class** <class\_name>{
2. field;
3. method;
4. }

Một biến được tạo bên trong lớp nhưng bên ngoài phương thức được gọi là biến thể hiện. Biến thể hiện không nhận được bộ nhớ tại thời điểm biên dịch. Nó nhận bộ nhớ trong thời gian chạy khi một đối tượng hoặc thể hiện được tạo. Đó là lý do tại sao nó được gọi là một biến thể hiện.

Trong Java, một phương thức giống như một hàm được sử dụng để hiển thị hành vi của một đối tượng.

Lợi thế của phương pháp

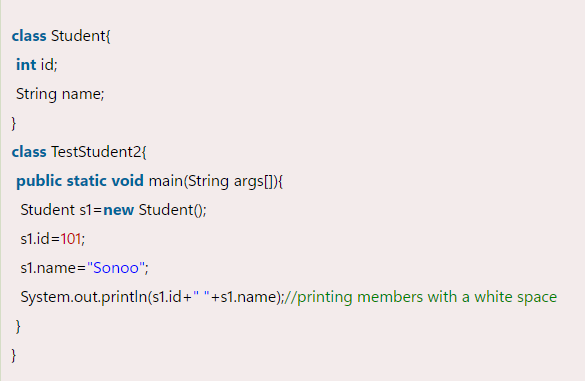
* Khả năng sử dụng lại mã
* Tối ưu hóa mã

Từ khóa mới được sử dụng để phân bổ bộ nhớ trong thời gian chạy. Tất cả các đối tượng lấy bộ nhớ trong vùng bộ nhớ Heap.

Có 3 cách để khởi tạo đối tượng trong Java.

1. Theo biến tham chiếu
2. Bằng phương pháp
3. Theo nhà xây dựng

 Ví dụ về đối tượng và lớp: Khởi tạo thông qua tham chiếu



Hình I.1: Đoạn code ví dụ về khởi tạo thông qua tham chiếu



*Hình I.2: Kết quả đoạn code trên.*

1. **Inherit**

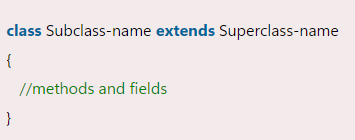
**Kế thừa trong Java** là một cơ chế trong đó một đối tượng có được tất cả các thuộc tính và hành vi của đối tượng cha. Nó là một phần quan trọng của [OOPs](https://www.javatpoint.com/java-oops-concepts) (Hệ thống lập trình hướng đối tượng).

Ý tưởng đằng sau tính kế thừa trong Java là bạn có thể tạo [các lớp](https://www.javatpoint.com/object-and-class-in-java) mới được xây dựng dựa trên các lớp hiện có. Khi bạn kế thừa từ một lớp hiện có, bạn có thể sử dụng lại các phương thức và trường của lớp cha. Hơn nữa, bạn cũng có thể thêm các phương thức và trường mới trong lớp hiện tại của mình.

Kế thừa đại diện cho **mối quan hệ IS-A** còn được gọi là mối quan hệ cha-con .

* **Lớp:** Một lớp là một nhóm các đối tượng có các thuộc tính chung. Nó là một mẫu hoặc bản thiết kế mà từ đó các đối tượng được tạo ra.
* **Lớp con/Lớp con:** Lớp con là lớp kế thừa lớp kia. Nó còn được gọi là lớp dẫn xuất, lớp mở rộng hoặc lớp con.
* **Lớp cấp cao/Lớp cha:** Lớp cấp trên là lớp mà từ đó một lớp con kế thừa các tính năng. Nó còn được gọi là lớp cơ sở hay lớp cha.
* **Khả năng sử dụng lại:** Như tên gọi, khả năng sử dụng lại là một cơ chế tạo điều kiện cho bạn sử dụng lại các trường và phương thức của lớp hiện có khi bạn tạo một lớp mới. Bạn có thể sử dụng cùng các trường và phương thức đã được định nghĩa trong lớp trước.

Cú pháp của Kế thừa Java



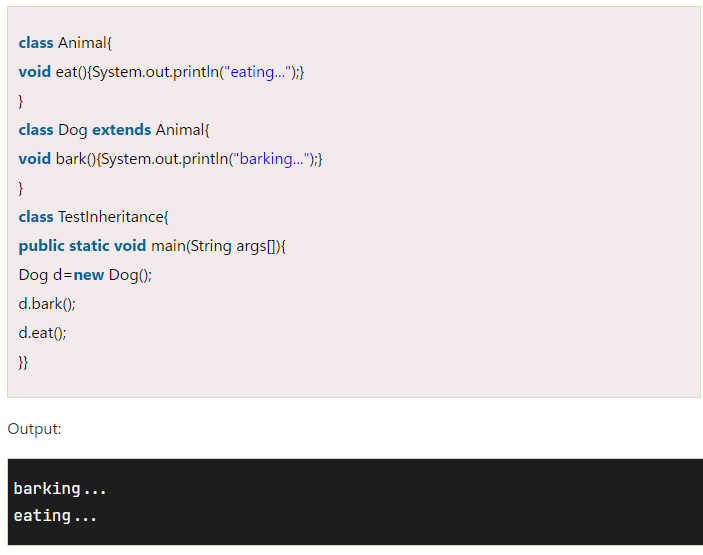
*Hình II.1: Cú pháp khai báo kế thừa Java*

Từ khóa **extends** chỉ ra rằng bạn đang tạo một lớp mới bắt nguồn từ một lớp hiện có. Ý nghĩa của "mở rộng" là tăng chức năng.

Theo thuật ngữ của Java, một lớp được kế thừa được gọi là lớp cha hoặc lớp cha và lớp mới được gọi là lớp con hoặc lớp con.

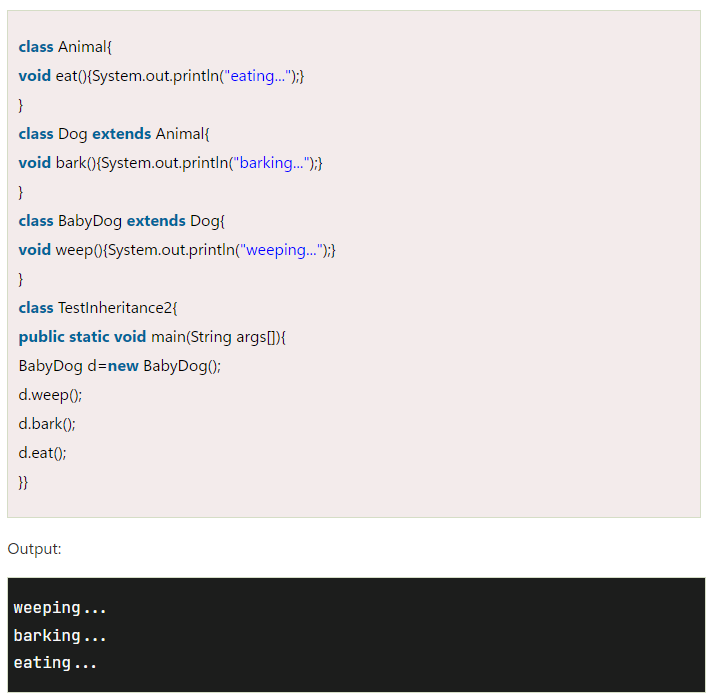
Trên cơ sở lớp, có thể có ba loại kế thừa trong java: đơn, đa cấp và phân cấp.

Ví dụ về kế thừa đơn:



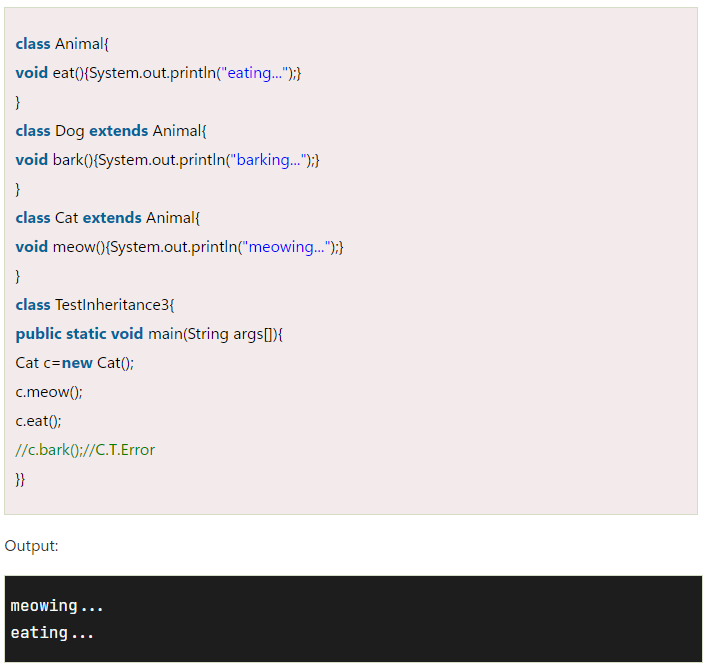
*Hình II.2: Code minh họa kế thừa đơn*

Ví dụ về kế thừa đa cấp:



Hình II.3: Code minh họa kế thừa đa cấp

Ví dụ về kế thừa phân cấp:



*Hình II.4: Code minh họa về kế thừa phân cấp*

1. **Polymorphism**

**Tính đa hình trong Java** là một khái niệm mà nhờ đó chúng ta có thể thực hiện một hành động đơn lẻ theo nhiều cách khác nhau . Đa hình có nguồn gốc từ 2 từ Hy Lạp: poly và morphs. Từ "poly" có nghĩa là nhiều và "morphs" có nghĩa là hình thức. Vì vậy, đa hình có nghĩa là nhiều hình thức.

Có hai loại đa hình trong Java: đa hình thời gian biên dịch và đa hình thời gian chạy. Chúng ta có thể thực hiện tính đa hình trong java bằng nạp chồng phương thức và ghi đè phương thức.

Nếu bạn quá tải một phương thức tĩnh trong Java, thì đó là ví dụ về đa hình thời gian biên dịch. Ở đây, chúng ta sẽ tập trung vào tính đa hình thời gian chạy trong java.

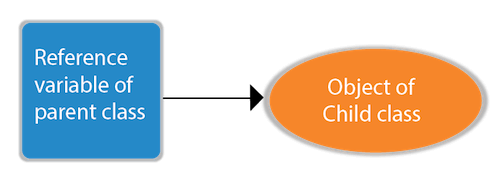
**Tính đa hình trong thời gian chạy** hoặc **Công văn phương thức động** là một quá trình trong đó lệnh gọi đến một phương thức bị ghi đè được giải quyết trong thời gian chạy thay vì thời gian biên dịch.

Trong quá trình này, một phương thức bị ghi đè được gọi thông qua biến tham chiếu của một lớp cha. Việc xác định phương thức được gọi dựa trên đối tượng được tham chiếu bởi biến tham chiếu.

Trước tiên chúng ta hãy hiểu về upcasting trước Đa hình thời gian chạy.

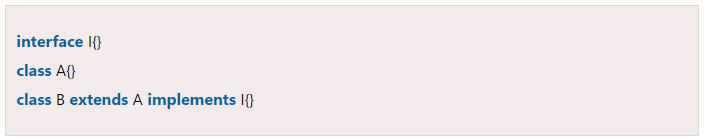
Upcasting

Nếu biến tham chiếu của lớp Cha đề cập đến đối tượng của lớp Con, nó được gọi là upcasting. Ví dụ:





Để upcasting, chúng ta có thể sử dụng biến tham chiếu của loại lớp hoặc loại giao diện. Ví dụ:



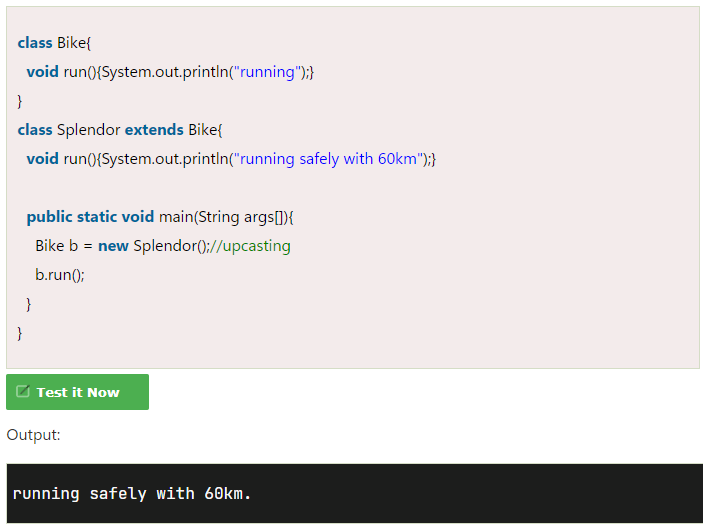
Ở đây, mối quan hệ của lớp B sẽ là:



Ví dụ về tính đa hình trong thời gian chạy Java:

Trong ví dụ này, chúng ta đang tạo hai lớp Bike và Splendor. Lớp Splendor mở rộng lớp Bike và ghi đè phương thức run() của nó. Chúng ta đang gọi phương thức chạy bằng biến tham chiếu của lớp Parent. Vì nó đề cập đến đối tượng của lớp con và phương thức của lớp con sẽ ghi đè lên phương thức của lớp Cha, nên phương thức của lớp con được gọi trong thời gian chạy.

Vì lời gọi phương thức được xác định bởi trình biên dịch JVM chứ không phải trình biên dịch, nên nó được gọi là đa hình thời gian chạy.



Hình III.1: Code minh họa về tính đa hình trong thời gian chạy Java

1. **Abstract**

Một lớp được khai báo với từ khóa trừu tượng được gọi là lớp trừu tượng trong [Java](https://www.javatpoint.com/java-tutorial) . Nó có thể có các phương thức trừu tượng và không trừu tượng (phương thức có phần thân).

Trước khi tìm hiểu về lớp trừu tượng trong Java, trước tiên chúng ta hãy hiểu về tính trừu tượng trong Java.

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

Một cách khác, nó chỉ hiển thị những thứ cần thiết cho người dùng và ẩn các chi tiết bên trong, chẳng hạn như gửi SMS khi bạn nhập văn bản và gửi tin nhắn. Bạn không biết quá trình xử lý nội bộ về việc gửi tin nhắn.

Tính trừu tượng cho phép bạn tập trung vào những gì [đối tượng](https://www.javatpoint.com/object-and-class-in-java) thực hiện thay vì cách thức thực hiện.

Các cách để đạt được Trừu tượng

Có hai cách để đạt được sự trừu tượng hóa trong java

1. Lớp trừu tượng (0 đến 100%)
2. Giao diện (100%)

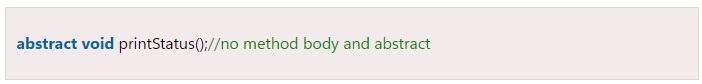
Một lớp được khai báo là trừu tượng được gọi là lớp trừu tượng . Nó có thể có các phương thức trừu tượng và không trừu tượng. Nó cần được mở rộng và phương pháp của nó được thực hiện. Nó không thể được khởi tạo.

Những điểm cần nhớ

* Một lớp trừu tượng phải được khai báo bằng từ khóa trừu tượng.
* Nó có thể có các phương thức trừu tượng và không trừu tượng.
* Nó không thể được khởi tạo.
* Nó có thể có [các hàm tạo](https://www.javatpoint.com/java-constructor) và các phương thức tĩnh.
* Nó có thể có các phương thức cuối cùng sẽ buộc lớp con không thay đổi phần thân của phương thức.

Một phương thức được khai báo là trừu tượng và không có triển khai được gọi là phương thức trừu tượng.

**Ví dụ về phương pháp trừu tượng**



*Hình IV.1: Code minh họa phương pháp trừu tượng trong Java*

1. **Encapsulation**

Đóng gói trong Java là một quá trình gói mã và dữ liệu lại với nhau thành một đơn vị duy nhất , ví dụ: một viên nang chứa nhiều loại thuốc.

Chúng ta có thể tạo một lớp được đóng gói đầy đủ trong Java bằng cách đặt tất cả các thành viên dữ liệu của lớp ở chế độ riêng tư. Bây giờ chúng ta có thể sử dụng các phương thức setter và getter để thiết lập và lấy dữ liệu trong đó.

Lớp Java Bean là ví dụ về một lớp được đóng gói đầy đủ.

Lợi thế của việc đóng gói trong Java

Bằng cách chỉ cung cấp một phương thức setter hoặc getter, bạn có thể làm cho lớp chỉ đọc hoặc chỉ ghi . Nói cách khác, bạn có thể bỏ qua các phương thức getter hoặc setter.

Nó cung cấp cho bạn quyền kiểm soát dữ liệu . Giả sử bạn muốn đặt giá trị của id chỉ lớn hơn 100, bạn có thể viết logic bên trong phương thức setter. Bạn có thể viết logic để không lưu trữ các số âm trong các phương thức setter.

Đó là một cách để ẩn dữ liệu trong Java vì lớp khác sẽ không thể truy cập dữ liệu thông qua các thành viên dữ liệu riêng tư.

Lớp đóng gói rất dễ kiểm tra . Vì vậy, nó là tốt hơn cho thử nghiệm đơn vị.

IDE tiêu chuẩn đang cung cấp phương tiện để tạo getters và setters. Vì vậy, thật dễ dàng và nhanh chóng để tạo một lớp đóng gói trong Java.\

1. **Object**

Lớp Object mặc định là lớp cha của tất cả các lớp trong java. Nói cách khác, nó là lớp trên cùng của java.

Lớp Object rất hữu ích nếu bạn muốn tham chiếu bất kỳ đối tượng nào có kiểu mà bạn không biết. Lưu ý rằng biến tham chiếu lớp cha có thể tham chiếu đối tượng lớp con, được gọi là upcasting.

Hãy lấy một ví dụ, có phương thức getObject() trả về một đối tượng nhưng nó có thể thuộc bất kỳ loại nào như Nhân viên, Sinh viên, v.v., chúng ta có thể sử dụng tham chiếu lớp Đối tượng để chỉ đối tượng đó.

Lớp Object cung cấp một số hành vi chung cho tất cả các đối tượng như đối tượng có thể được so sánh, đối tượng có thể được nhân bản, đối tượng có thể được thông báo, v.v.

|  |  |
| --- | --- |
| Phương pháp | Sự miêu tả |
| lớp cuối cùng công khai getClass() | trả về đối tượng lớp Class của đối tượng này. Lớp Lớp có thể được sử dụng thêm để lấy siêu dữ liệu của lớp này. |
| public int hashCode() | trả về số mã băm cho đối tượng này. |
| boolean công khai bằng (Đối tượng obj) | so sánh đối tượng đã cho với đối tượng này. |
| bản sao đối tượng được bảo vệ () ném CloneNotSupportedException | tạo và trả về bản sao chính xác (bản sao) của đối tượng này. |
| chuỗi công khai toString() | trả về biểu diễn chuỗi của đối tượng này. |
| công khai khoảng trống cuối cùng thông báo() | đánh thức chuỗi đơn, chờ trên màn hình của đối tượng này. |
| khoảng trống cuối cùng công khai thông báo cho tất cả () | đánh thức tất cả các chủ đề, chờ đợi trên màn hình của đối tượng này. |
| public final void wait(long timeout) throws InterruptedException | làm cho luồng hiện tại đợi trong một phần nghìn giây đã chỉ định, cho đến khi một luồng khác thông báo (gọi phương thức notify() hoặc notifyAll()). |
| đợi khoảng trống cuối cùng công khai (thời gian chờ dài, int nanos) ném InterruptedException | làm cho luồng hiện tại đợi trong mili giây và nano giây đã chỉ định, cho đến khi luồng khác thông báo (gọi phương thức notify() hoặc notifyAll()). |
| public final void wait() throws InterruptedException | làm cho luồng hiện tại đợi, cho đến khi luồng khác thông báo (gọi phương thức notify() hoặc notifyAll()). |
| protected void finalize() throws Ném được | được gọi bởi trình thu gom rác trước khi đối tượng được thu gom rác. |

1. **Math**

Lớp Java Math cung cấp một số phương thức để thực hiện các phép tính toán học như min(), max(), avg(), sin(), cos(), tan(), round(), ceil(), floor(), abs( ) vân vân.

Không giống như một số phương thức số của lớp StrictMath, tất cả các triển khai chức năng tương đương của lớp Math không thể xác định để trả về kết quả giống nhau từng bit. Sự thư giãn này cho phép thực hiện với hiệu suất tốt hơn khi không yêu cầu khả năng tái tạo nghiêm ngặt.

Nếu kích thước là int hoặc long và kết quả vượt quá phạm vi giá trị, thì các phương thức addExact(),trừExact(),multiExact() và toIntExact() sẽ đưa ra Ngoại lệ Số học.

Đối với các phép toán số học khác như tăng, giảm, chia, giá trị tuyệt đối và tràn phủ định chỉ xảy ra với một giá trị tối thiểu hoặc tối đa cụ thể. Nó nên được kiểm tra đối với giá trị tối đa và tối thiểu khi thích hợp.

Lớp java.lang.Math chứa các phương thức khác nhau để thực hiện các phép tính số cơ bản như logarit, căn bậc ba và các hàm lượng giác, v.v. Các phương thức toán học java khác nhau như sau:

Phương pháp toán học cơ bản

|  |  |
| --- | --- |
| Phương pháp | Sự miêu tả |
| [Math.abs()](https://www.javatpoint.com/java-math-abs-method) | Nó sẽ trả về giá trị Tuyệt đối của giá trị đã cho. |
| [Math.max()](https://www.javatpoint.com/java-math-max-method) | Nó trả về Giá trị lớn nhất trong hai giá trị. |
| [Math.min()](https://www.javatpoint.com/java-math-min-method) | Nó được sử dụng để trả về Giá trị nhỏ nhất trong hai giá trị. |
| [Math.round()](https://www.javatpoint.com/java-math-round-method) | Nó được sử dụng để làm tròn các số thập phân đến giá trị gần nhất. |
| [Math.sqrt()](https://www.javatpoint.com/java-math-sqrt-method) | Nó được sử dụng để trả về căn bậc hai của một số. |
| [Math.cbrt()](https://www.javatpoint.com/java-math-cbrt-method) | Nó được sử dụng để trả về căn bậc ba của một số. |
| [Math.pow()](https://www.javatpoint.com/java-math-pow-method) | Nó trả về giá trị của đối số đầu tiên được nâng lên lũy thừa thành đối số thứ hai. |
| [Math.signum()](https://www.javatpoint.com/java-math-signum-method) | Nó được sử dụng để tìm dấu của một giá trị nhất định. |
| [Math.ceil()](https://www.javatpoint.com/java-math-ceil-method) | Nó được sử dụng để tìm giá trị số nguyên nhỏ nhất lớn hơn hoặc bằng đối số hoặc số nguyên toán học. |
| [Math.copySign()](https://www.javatpoint.com/java-math-copysign-method) | Nó được sử dụng để tìm giá trị Tuyệt đối của đối số đầu tiên cùng với dấu được chỉ định trong đối số thứ hai. |
| [Math.nextSau()](https://www.javatpoint.com/java-math-nextafter-method) | Nó được sử dụng để trả về số dấu phẩy động liền kề với đối số thứ nhất theo hướng của đối số thứ hai. |
| [Math.nextUp()](https://www.javatpoint.com/java-math-nextup-method) | Nó trả về giá trị dấu phẩy động liền kề với d theo hướng vô cực dương. |
| [Math.nextDown()](https://www.javatpoint.com/java-math-nextdown-method) | Nó trả về giá trị dấu phẩy động liền kề với d theo hướng vô cực âm. |
| [Math.sàn ()](https://www.javatpoint.com/java-math-floor-method) | Nó được sử dụng để tìm giá trị số nguyên lớn nhất nhỏ hơn hoặc bằng đối số và bằng số nguyên toán học của một giá trị kép. |
| [Math.floorDiv()](https://www.javatpoint.com/java-math-floordiv-method) | Nó được sử dụng để tìm giá trị số nguyên lớn nhất nhỏ hơn hoặc bằng thương số đại số. |
| [Math.random()](https://www.javatpoint.com/java-math-random-method) | Nó trả về một giá trị kép có dấu dương, lớn hơn hoặc bằng 0,0 và nhỏ hơn 1,0. |
| [Math.rint()](https://www.javatpoint.com/java-math-rint-method) | Nó trả về giá trị kép gần nhất với đối số đã cho và bằng số nguyên toán học. |
| [Math.hypot()](https://www.javatpoint.com/java-math-hypot-method) | Nó trả về sqrt(x 2  +y 2 ) mà không có tràn hoặc tràn trung gian. |
| [Math.ulp()](https://www.javatpoint.com/java-math-ulp-method) | Nó trả về kích thước của một ulp của đối số. |
| [Math.getExponent()](https://www.javatpoint.com/java-math-getexponent-method) | Nó được sử dụng để trả về số mũ không thiên vị được sử dụng trong biểu diễn của một giá trị. |
| [Math.IEEEremainder()](https://www.javatpoint.com/java-math-ieeeremainder-method) | Nó được sử dụng để tính toán phép toán còn lại trên hai đối số theo quy định của tiêu chuẩn IEEE 754 và trả về giá trị. |
| [Math.addExact()](https://www.javatpoint.com/java-math-addexact-method) | Nó được sử dụng để trả về tổng các đối số của nó, đưa ra một ngoại lệ nếu kết quả vượt quá một int hoặc long. |
| [Math.subtractExact()](https://www.javatpoint.com/java-math-subtractexact-method) | Nó trả về sự khác biệt của các đối số, đưa ra một ngoại lệ nếu kết quả vượt quá một int. |
| [Math.multiplyExact()](https://www.javatpoint.com/java-math-multiplyexact-method) | Nó được sử dụng để trả về tích của các đối số, đưa ra một ngoại lệ nếu kết quả vượt quá một int hoặc long. |
| [Math.incrementExact()](https://www.javatpoint.com/java-math-incrementexact-method) | Nó trả về đối số tăng thêm một, đưa ra một ngoại lệ nếu kết quả vượt quá một int. |
| [Math.decrementExact()](https://www.javatpoint.com/java-math-decrementexact-method) | Nó được sử dụng để trả về đối số giảm đi một, đưa ra một ngoại lệ nếu kết quả vượt quá một int hoặc long. |
| [Math.negateExact()](https://www.javatpoint.com/java-math-negateexact-method) | Nó được sử dụng để trả về phủ định của đối số, đưa ra một ngoại lệ nếu kết quả vượt quá một int hoặc long. |
| [Math.toIntExact()](https://www.javatpoint.com/java-math-tointexact-method) | Nó trả về giá trị của đối số dài, đưa ra một ngoại lệ nếu giá trị vượt quá một int. |

Phương pháp toán logarit

|  |  |
| --- | --- |
| Phương pháp | Sự miêu tả |
| [Math.log()](https://www.javatpoint.com/java-math-log-method) | Nó trả về logarit tự nhiên của một giá trị kép. |
| [Math.log10()](https://www.javatpoint.com/java-math-log10-method) | Nó được sử dụng để trả về logarit cơ số 10 của một giá trị kép. |
| [Math.log1p()](https://www.javatpoint.com/java-math-log1p-method) | Nó trả về logarit tự nhiên của tổng đối số và 1. |
| [Math.exp()](https://www.javatpoint.com/java-math-exp-method) | Nó trả về E lũy thừa của một giá trị kép, trong đó E là số Euler và nó xấp xỉ bằng 2,71828. |
| [Math.expm1()](https://www.javatpoint.com/java-math-expm1-method) | Nó được sử dụng để tính toán sức mạnh của E và trừ đi một từ nó. |

Phương pháp toán lượng giác

|  |  |
| --- | --- |
| Phương pháp | Sự miêu tả |
| [Math.sin()](https://www.javatpoint.com/java-math-sin-method) | Nó được sử dụng để trả về giá trị Sine lượng giác của một giá trị kép đã cho. |
| [Math.cos()](https://www.javatpoint.com/java-math-cos-method) | Nó được sử dụng để trả về giá trị Cosine lượng giác của một giá trị kép đã cho. |
| [Math.tan()](https://www.javatpoint.com/java-math-tan-method) | Nó được sử dụng để trả về giá trị Tangent lượng giác của một giá trị kép đã cho. |
| [Math.asin()](https://www.javatpoint.com/java-math-asin-method) | Nó được sử dụng để trả về giá trị Arc Sin lượng giác của một giá trị kép đã cho |
| [Math.acos()](https://www.javatpoint.com/java-math-acos-method) | Nó được sử dụng để trả về giá trị Arc Cosine lượng giác của một giá trị kép đã cho. |
| [Math.atan()](https://www.javatpoint.com/java-math-atan-method) | Nó được sử dụng để trả về giá trị Tiếp tuyến cung lượng giác của một giá trị kép đã cho. |

Phương pháp Toán Hyperbolic

|  |  |
| --- | --- |
| Phương pháp | Sự miêu tả |
| [Math.sinh()](https://www.javatpoint.com/java-math-sinh-method) | Nó được sử dụng để trả về giá trị Hyperbolic Cosine lượng giác của một giá trị kép đã cho. |
| [Math.cosh()](https://www.javatpoint.com/java-math-cosh-method) | Nó được sử dụng để trả về giá trị sin Hyperbolic lượng giác của một giá trị kép đã cho. |
| [Math.tanh()](https://www.javatpoint.com/java-math-tanh-method) | Nó được sử dụng để trả về giá trị Tangent Hyperbolic lượng giác của một giá trị kép đã cho. |

Phương pháp toán góc

|  |  |
| --- | --- |
| Phương pháp | Sự miêu tả |
| [Math.toDegrees](https://www.javatpoint.com/java-math-todegrees-method) | Nó được sử dụng để chuyển đổi góc Radian đã chỉ định thành góc tương đương được đo bằng Độ. |
| [Math.toRadians](https://www.javatpoint.com/java-math-toradians-method) | Nó được sử dụng để chuyển đổi góc Độ đã chỉ định thành góc tương đương được đo bằng Radian. |

1. **Wrapper**

Lớp **trình bao bọc trong Java** cung cấp cơ chế để chuyển đổi nguyên thủy thành đối tượng và đối tượng thành nguyên thủy .

Kể từ J2SE 5.0, tính năng **autoboxing** và **unboxing** tự động chuyển đổi nguyên thủy thành đối tượng và đối tượng thành nguyên thủy. Việc tự động chuyển đổi nguyên thủy thành một đối tượng được gọi là autoboxing và ngược lại unboxing.

Java là ngôn ngữ lập trình hướng đối tượng, vì vậy chúng ta cần xử lý nhiều lần với các đối tượng như trong Bộ sưu tập, Tuần tự hóa, Đồng bộ hóa, v.v. Hãy để chúng tôi xem các tình huống khác nhau, nơi chúng tôi cần sử dụng các lớp bao bọc.

* **Thay đổi giá trị trong Phương thức:** Java chỉ hỗ trợ gọi theo giá trị. Vì vậy, nếu chúng ta chuyển một giá trị nguyên thủy, nó sẽ không thay đổi giá trị ban đầu. Nhưng, nếu chúng ta chuyển đổi giá trị nguyên thủy trong một đối tượng, nó sẽ thay đổi giá trị ban đầu.
* **Tuần tự hóa:** Chúng ta cần chuyển đổi các đối tượng thành luồng để thực hiện tuần tự hóa. Nếu chúng ta có một giá trị nguyên thủy, chúng ta có thể chuyển đổi nó trong các đối tượng thông qua các lớp bao bọc.
* **Đồng bộ hóa:** Đồng bộ hóa Java hoạt động với các đối tượng trong Đa luồng.
* **Gói java.util:** Gói java.util cung cấp các lớp tiện ích để xử lý các đối tượng.
* **Khung bộ sưu tập:** Khung bộ sưu tập Java chỉ hoạt động với các đối tượng. Tất cả các lớp của khung bộ sưu tập (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, v.v.) chỉ xử lý các đối tượng.

Tám lớp của gói java.lang được gọi là các lớp bao bọc trong Java. Danh sách tám lớp bao bọc được đưa ra dưới đây:

|  |  |
| --- | --- |
| Loại nguyên thủy | lớp bao bọc |
| boolean | [Boolean](https://www.javatpoint.com/java-boolean) |
| than | [Tính cách](https://www.javatpoint.com/post/java-character) |
| byte | [byte](https://www.javatpoint.com/java-byte) |
| ngắn | [Ngắn](https://www.javatpoint.com/java-short) |
| int | [số nguyên](https://www.javatpoint.com/java-integer) |
| dài | [Dài](https://www.javatpoint.com/java-long) |
| trôi nổi | [Trôi nổi](https://www.javatpoint.com/java-float) |
| gấp đôi | [Gấp đôi](https://www.javatpoint.com/java-double) |

Việc tự động chuyển đổi kiểu dữ liệu nguyên thủy thành lớp trình bao bọc tương ứng của nó được gọi là autoboxing, ví dụ: byte thành Byte, char thành Character, int thành Integer, long thành Long, float thành Float, boolean thành Boolean, double thành Double và short thành Ngắn gọn.

Kể từ Java 5, chúng ta không cần sử dụng phương thức valueOf() của các lớp bao bọc để chuyển đổi nguyên hàm thành các đối tượng.

1. **String buffer**

Lớp StringBuffer trong Java được sử dụng để tạo các đối tượng Chuỗi có thể thay đổi (có thể sửa đổi). Lớp StringBuffer trong Java cũng giống như lớp String ngoại trừ nó có thể thay đổi được tức là nó có thể thay đổi được.

### Các Constructor quan trọng của StringBuffer Class

|  |  |
| --- | --- |
| Người xây dựng | Sự miêu tả |
| Bộ đệm chuỗi () | Nó tạo một bộ đệm Chuỗi trống với dung lượng ban đầu là 16. |
| StringBuffer(Chuỗi str) | Nó tạo một bộ đệm Chuỗi với chuỗi đã chỉ định.. |
| StringBuffer(dung lượng int) | Nó tạo ra một bộ đệm Chuỗi trống với dung lượng được chỉ định là độ dài. |

### Các phương thức quan trọng của lớp StringBuffer

|  |  |  |
| --- | --- | --- |
| Công cụ sửa đổi và Loại | Phương pháp | Sự miêu tả |
| StringBuffer được đồng bộ hóa công khai | nối thêm (Chuỗi s) | Nó được sử dụng để nối chuỗi đã chỉ định với chuỗi này. Phương thức append() bị quá tải như append(char), append(boolean), append(int), append(float), append(double) v.v. |
| StringBuffer được đồng bộ hóa công khai | chèn (int offset, Chuỗi s) | Nó được sử dụng để chèn chuỗi đã chỉ định với chuỗi này vào vị trí đã chỉ định. Phương thức insert() bị quá tải như insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double), v.v. |
| StringBuffer được đồng bộ hóa công khai | thay thế (int start Index, int end Index, String str) | Nó được sử dụng để thay thế chuỗi từ start Index và end Index đã chỉ định. |
| StringBuffer được đồng bộ hóa công khai | xóa(int start Index, int end Index) | Nó được sử dụng để xóa chuỗi khỏi start Index và end Index đã chỉ định. |
| StringBuffer được đồng bộ hóa công khai | đảo ngược() | được sử dụng để đảo ngược chuỗi. |
| int công cộng | dung tích() | Nó được sử dụng để trả về dung lượng hiện tại. |
| khoảng trống công cộng | ensureCapacity(int minimumCapacity) | Nó được sử dụng để đảm bảo dung lượng ít nhất bằng mức tối thiểu đã cho. |
| công khai | charAt(chỉ số int) | Nó được sử dụng để trả về ký tự tại vị trí đã chỉ định. |
| int công cộng | chiều dài() | Nó được sử dụng để trả về độ dài của chuỗi tức là tổng số ký tự. |
| chuỗi công khai | chuỗi con (int bắt đầu Index) | Nó được sử dụng để trả về chuỗi con từ beginIndex đã chỉ định. |
| chuỗi công khai | chuỗi con (int start Index, int end Index) | Nó được sử dụng để trả về chuỗi con từ start Index và end Index đã chỉ định. |

Chuỗi có thể sửa đổi hoặc thay đổi được gọi là Chuỗi có thể thay đổi. Các lớp StringBuffer và StringBuilder được sử dụng để tạo các chuỗi có thể thay đổi.

* Phương thức append() của lớp StringBuffer
* Phương thức chèn StringBuffer()
* Phương thức thay thế StringBuffer()
* Phương thức xóa StringBuffer()
* Phương thức đảo ngược StringBuffer()
* Dung lượng StringBuffer() Phương thức
* Phương thức StringBuffer ensureCapacity()

1. **This**

Có thể có rất nhiều cách sử dụng **this** . Trong Java, đây là một **biến tham chiếu** đề cập đến đối tượng hiện tại.

Dưới đây là 6 cách sử dụng java từ khóa này.

1. [điều này có thể được sử dụng để chỉ biến thể hiện của lớp hiện tại.](https://www.javatpoint.com/this1)
2. [điều này có thể được sử dụng để gọi phương thức lớp hiện tại (ngầm)](https://www.javatpoint.com/this2)
3. [this() có thể được sử dụng để gọi hàm tạo của lớp hiện tại.](https://www.javatpoint.com/this3)
4. [điều này có thể được chuyển thành đối số trong lệnh gọi phương thức.](https://www.javatpoint.com/this4)
5. [điều này có thể được chuyển thành đối số trong lệnh gọi hàm tạo.](https://www.javatpoint.com/this5)
6. [điều này có thể được sử dụng để trả về thể hiện của lớp hiện tại từ phương thức.](https://www.javatpoint.com/this6)
7. **Super**

Từ khóa super trong Java là một biến tham chiếu được sử dụng để chỉ đối tượng lớp cha ngay lập tức.

Bất cứ khi nào bạn tạo thể hiện của lớp con, một thể hiện của lớp cha được tạo hoàn toàn và được gọi bằng biến siêu tham chiếu.

Sử dụng siêu từ khóa Java

1. super có thể được sử dụng để chỉ biến thể hiện của lớp cha ngay lập tức.
2. super có thể được sử dụng để gọi phương thức của lớp cha ngay lập tức.
3. super() có thể được sử dụng để gọi hàm tạo của lớp cha ngay lập tức.