**Khái niệm ép kiểu**

* Ép kiểu là hành động đưa giá trị của một kiểu sang một kiểu khác.
* Kiểu ở đây là các kiểu dữ liệu hợp lệ trong ngôn ngữ Java như các kiểu nguyên thủy, tham chiếu.
* Ví dụ ta ép kiểu giá trị kiểu:
  + int sang float
  + float sang int
  + double sang float
  + double sang int
* Hành động ép kiểu có thể thể hiện tường minh, tức chỉ rõ kiểu đích cần ép sang. Hoặc ép kiểu ngầm định, tức không cần chỉ rõ kiểu đích, chương trình tự biết thực hiện.

**Lý do phải ép kiểu**

* Trong một ngữ cảnh nào đó, ta sẽ cần một kiểu dữ liệu cụ thể để thao tác được chính xác. Để đảm bảo điều này có thể thực hiện được, ta cần phải ép kiểu.
* Ví dụ:
  + Để có giá trị chính xác của phép chia 2/3 ta phải ép kiểu tử hoặc mẫu số sang kiểu số thực float hoặc double.
  + Để lấy phần nguyên của giá trị 3.14159 ta ép kiểu nó sang số nguyên kiểu int chẳng hạn.
  + Khi muốn các giá trị nguyên thủy có thể thao tác như các đối tượng, ta ép kiểu sang kiểu lớp bao của nó. Ví dụ: Integer intObj = 200;

**Có những cách ép kiểu nào**

* Có nhiều cách ép kiểu khác nhau. Chúng ta tập trung vào ép kiểu các kiểu nguyên thủy trước:
  + Ép sang cùng kiểu
  + Ép sang các kiểu rộng hơn
  + Ép sang các kiểu hẹp hơn
  + Ép sang kiểu rộng hơn sau đó sang kiểu hẹp hơn
  + Ép kiểu đóng hộp(boxing)
  + Ép kiểu mở hộp(unboxing)
* Ngoài ra còn có các thao tác ép kiểu nâng cao khác sẽ tìm hiểu ở các bài học tương ứng về sau. Ví dụ ép kiểu với chuỗi kí tự(String), ép kiểu với các kiểu tham chiếu…

**Ép sang cùng kiểu:**

* Đây là cách ép kiểu hay còn gọi chuyển đổi phổ thông nhất. Bạn có thể gán giá trị của các biến cùng kiểu cho nhau. Hành động này không xảy ra lỗi nào cả. Ép kiểu sang giá trị cùng kiểu mặc định là ép kiểu ngầm định. Ta vẫn có thể thể hiện ép kiểu cách này một cách tường minh nếu muốn.
* Khi ép kiểu theo cách này, dữ liệu gốc được bảo toàn.
* Ví dụ:

int a = 200;

int b = a; *// ok – ép kiểu ngầm định*

int c = (int)b; *// ok – ép kiểu tường minh*

**Ép sang các kiểu rộng hơn:**

* Mặc định khi ta ép kiểu từ kiểu có kích thước nhỏ hơn sang kiểu có kích thước lớn hơn hoặc từ kiểu con sang kiểu cha thì nó thuộc loại ép kiểu này.
* Các thao tác ép kiểu nhóm này sẽ thực hiện một cách ngầm định. Không cần phải chỉ rõ kiểu đích.
* Khi ép kiểu theo cách này, dữ liệu gốc được bảo toàn. Các trường hợp thỏa mãn là:
  + Ép từ kiểu nguyên nhỏ hơn sang kiểu nguyên lớn hơn
  + Ép từ byte, short, char sang kiểu số thực
  + Ép từ int sang double
  + Từ float sang double ở biểu thức strictfp
* Ví dụ ép kiểu:
  + Ép từ kiểu byte sang short, int, long, float, double
  + Ép từ kiểu short sang kiểu int, long, float, double
  + Ép kiểu từ int sang long, float hoặc double
  + Ép kiểu float sang double
* Ví dụ chứng minh mất dữ liệu khi ép kiểu:

int a = 1234567890;

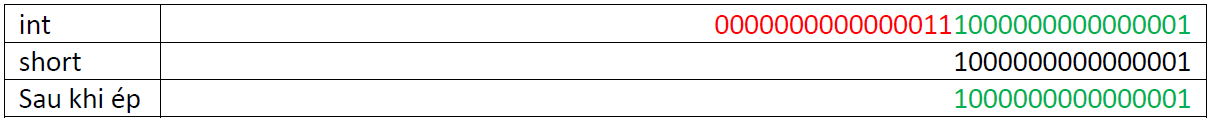
float f = a;

System.*out*.println(a - (int)f);// Kết quả -46

Ví dụ này chỉ ra rằng. Nếu việc ép kiểu từ kiểu nhỏ hơn sang kiểu lớn hơn mà thực hiện chính xác, thì việc ép kiểu ngược lại cũng sẽ cho cùng giá trị. Nhưng không phải tất cả mọi ép kiểu loại này đều đúng như vậy.

**Ép sang kiểu hẹp hơn**

* Ép kiểu loại này xảy ra khi thực hiện ép kiểu từ kiểu lớn hơn sang kiểu nhỏ hơn.
* Để ép kiểu thành công, ta sẽ phải thực hiện tường minh.
* Ép kiểu loại này sẽ gây mất dữ liệu. Đây là điều không thể tránh khỏi. Lý do mất dữ liệu là kiểu nhỏ hơn chỉ chứa được n bít thấp(màu xanh) của các bit kiểu gốc trước khi ép sang. Cụ thể hơn:

[](https://braniumacademy.net/wp-content/uploads/2020/08/l%C3%BD-do-%C3%A9p-ki%E1%BB%83u-m%E1%BA%A5t-d%E1%BB%AF-li%E1%BB%87u.png)

* Ví dụ:
  + Ép từ kiểu int sang kiểu short
  + Ép từ kiểu double sang float hoặc long, int..
* Cụ thể:

float f = 1.25f;

int newValue = (int)f;

System.*out*.println(newValue);

*// kết quả: 1*

**Ép sang kiểu rộng hơn sau đó sang kiểu hẹp hơn**

* Trường hợp này xảy ra khi ta ép kiểu byte sang char. Các bước thực hiện là:
  + Đầu tiên byte sẽ chuyển sang kiểu int(ép kiểu rộng)
  + Sau đó từ int sẽ ép xuống kiểu char(ép kiểu hẹp)

**Ép kiểu đóng hộp**

* Điều này xảy ra khi ta đưa giá trị nguyên thủy sang giá trị kiểu lớp bao tương ứng của nó.
* Ví dụ:
  + Ép từ kiểu boolean sang Boolean
  + Ép từ byte sang Byte
  + Ép từ short sang Short
  + Ép từ char sang Character
  + Ép từ int sang Integer
* Phương pháp hay cách ép kiểu này thường xuyên được áp dụng qua lại trong chương trình.
* Mục đích của việc ép kiểu theo phương pháp này là để sử dụng giá trị kiểu nguyên thủy như là một đối tượng. Từ đó có thể dễ dàng thực hiện các thao tác cần thiết như gọi phương thức, so sánh…

int x = 200; *// x là kiểu nguyên thủy*

Integer xObject = x; *// xObject là kiểu tham chiếu*

System.out.println(xObject.toString()); // ok

System.out.println(xObject.compareTo(200)); // ok

System.out.println(x.toString()); // error

System.out.println(x.compareTo(200)); // error

**Ép kiểu mở hộp**

* Khi thực hiện việc đưa giá trị của kiểu Boxing nào đó về kiểu nguyên thủy tương ứng. Nói cách khác, khi đưa giá trị của kiểu tham chiếu về kiểu nguyên thủy tương ứng, ta gọi là ép kiểu mở hộp(unboxing).
* Cùng với ép kiểu đóng hộp(boxing) thì ép kiểu mở hộp thường xuyên được thực hiện trong hệ thống của Java khi cần thực hiện các thao tác cần thiết.
* Ví dụ:
  + Ép từ Boolean sang kiểu boolean
  + Ép từ kiểu Integer sang int
  + Ép từ Float sang kiểu float
* Cụ thể:

Integer xObject = 500; *// xObject là kiểu tham chiếu*

int x = xObject.intValue(); *// x là kiểu nguyên thủy*

*// tương đương:*

x = xObject;

Double dObject = 123.456;

double d = dObject.doubleValue();

*// tương đương:*

d = dObject;

**Làm tròn số**

* Việc làm tròn số thường xuyên xảy ra với các giá trị số thực. Để làm tròn số trong Java có nhiều cách khác nhau:
  + Sử dụng phương thức Math.ceil() để làm tròn lên số nguyên gần nhất.
  + Sử dụng phương thức Math.floor() để làm tròn xuống số nguyên gần nhất.
  + Sử dụng phương thức Math.round() để làm tròn tự động lên hoặc xuống giá trị nguyên gần nhất. Quy ước mọi phần thập phân lẻ >= 0.5 sẽ làm tròn lên, ngược lại làm tròn xuống.
  + Làm tròn đến k chữ số với printf và %n.kf. Trong đó n là độ rộng tính theo số kí tự để in ra kết quả. Còn k là số chữ số cần làm tròn phần thập phân.
  + Sử dụng DecimalFormat. Trong đó mỗi chữ số sẽ được đánh dấu bởi dấu thăng phân tách phần nguyên, thập phân bởi dấu .
* Ví dụ:

double x = 2;

double y = 3;

double result = x / y; *// kết quả 0.6666666(6)*

System.*out*.println(Math.*ceil*(result)); *// kq = 1.0, lấy phần nguyên được 1*

System.*out*.println(Math.*floor*(result)); *// kq = 0.0, lấy phần nguyên được 0*

System.*out*.println(Math.*round*(result)); *// kq = 1*

*// sử dụng DecimalFormat*

NumberFormat numberFormat = new DecimalFormat("###.#########");

System.*out*.println(numberFormat.format(result));

*// sử dụng printf và chuỗi chuyển đổi định dạng:*

System.*out*.printf("%10.9f", result);

**Ví dụ minh họa**

import java.text.DecimalFormat;

import java.text.NumberFormat;

public class Lesson5 {

    public static void main(String[] args) {

        double x = 2;

        double y = 3;

        double result = x / y; *// kết quả 0.6666666(6)*

System.*out*.println(Math.*ceil*(result)); *// kq = 1*

System.*out*.println(Math.*floor*(result)); *// kq = 0*

System.*out*.println(Math.*round*(result)); *// kq = 1*

*// sử dụng DecimalFormat*

*// NumberFormat là kiểu cha của DecimalFormat*

NumberFormat numberFormat = new DecimalFormat("###.#########");

        System.*out*.println(numberFormat.format(result));

*// sử dụng printf và chuỗi chuyển đổi định dạng:*

System.*out*.printf("%10.9f", result);

    }

}