20 phút 1 câu hỏi Không giới hạn lượt làm lại 2 phút

Bài tập

Bạn hãy cho biết kết quả khi chạy chương trình sau:

```
class Math{
    public static int max(int a, int b) {
        return a > b ? a : b;
    }

    public static int max(int a, int b, int c) {
        return max(max(a, b), c);
    }
}

class Entry {
    public static void main(String[] args) {
        System.out.println(Math.max(3, 5, 2));
    }
}
```

Lý thuyết

Bài này sẽ giúp bạn hiểu về khái niệm nạp chồng phương thức trong Java.

Trước hết bạn hãy xem lại hàm `abs` của lớp `Math` trong bài trước:

```
class Math {
    public static int abs(int x) {
        return x < 0 ? -x : x;
    }
}
```

Vấn đề gặp phải với hàm này là tham số của hàm là một số nguyên, vậy nếu bạn muốn dùng hàm `abs` để tính giá trị tuyệt đối của một số thực hay một số nguyên kiểu `long` thì sao? Có thể bạn nghĩ ngay tới cách tạo thêm 2 hàm với tên gọi khác nhau như sau:

```
        return x < 0 ? -x : x;
    }
    public static double absDouble(double x) {
        return x < 0 ? -x : x;
    }
    public static long absLong(long x) {
        return x < 0 ? -x : x;
    }
}
```

Vấn đề với cách làm này là bạn cần phải tạo ra nhiều tên hàm cho các tham số khác nhau, tại sao đều là trị tuyệt đối mà không thể dùng chung tên hàm là `abs()`?

Trước hết bạn hãy thử gọi hàm `abs` của lớp `Math` có sẵn trong Java với các tham số kiểu `int, double, long`:

```
class Entry {
    public static void main(String[] args) {
        int a = -5;
        double b = -7;
        long c = -9;
        System.out.println(java.lang.Math.abs(a));
        System.out.println(java.lang.Math.abs(b));
        System.out.println(java.lang.Math.abs(c));
    }
}
```

Kết quả khi chạy chương trình:

```
5
7.0
9
```

Có thể thấy phương thức `abs()` của lớp `Math` có sẵn dùng được cho cả 3 kiểu dữ liệu là `int, float, double`. Kỹ thuật để lớp `Math` làm được việc này gọi là nạp chồng phương thức.

Nạp chồng phương thức

Nếu một lớp có nhiều phương thức cùng tên nhưng khác nhau về kiểu dữ liệu hoặc số các tham số, thì đó là nạp chồng phương thức.

Ví dụ bạn có thể tạo ra 2 hàm `add` để tính tổng số nguyên và số thực như sau:

```
class Math {
    public static int add(int a, int b) {
```

```
    public static double add(double a, double b) {
        return a + b;
    }
}

class Entry {
    public static void main(String[] args) {
        System.out.println(Math.add(2, 4));
        System.out.println(Math.add(3.5, 4.3));
    }
}
```

Kết quả khi chạy chương trình:

```
6
7.8
```

Có thể thấy bạn không cần tạo ra 2 hàm với tên khác nhau để tính tổng 2 số. Đây cũng là cách mà lớp `Math` tạo ra phương thức `abs()`. Source code của phương thức `abs()` trong lớp `Math` sẽ giống như sau:

```
class Math{
    public static int abs(int a) {
        return (a < 0) ? -a : a;
    }

    public static long abs(long a) {
        return (a < 0) ? -a : a;
    }

    public static double abs(double a) {
        return (a <= 0.0D) ? 0.0D - a : a;
    }
}
```

Ngoài ra bạn còn có thể nạp chồng phương thức bằng cách thay đổi số tham số của phương thức:

```
class Math {
    public static int max(int a, int b) {
        return (a > b) ? a : b;
```

```
        if(maxValue < b) {
            maxValue = b;
        }
        if(maxValue < c) {
            maxValue = c;
        }
        return maxValue;
    }
}

class Entry {
    public static void main(String[] args) {
        System.out.println(Math.max(4, 5));
        System.out.println(Math.max(4, 5, 7));
    }
}
```

Kết quả khi chạy chương trình:

```
5
7
```

Ngoài ra, các phương thức được nạp chồng còn có thể gọi tới nhau:

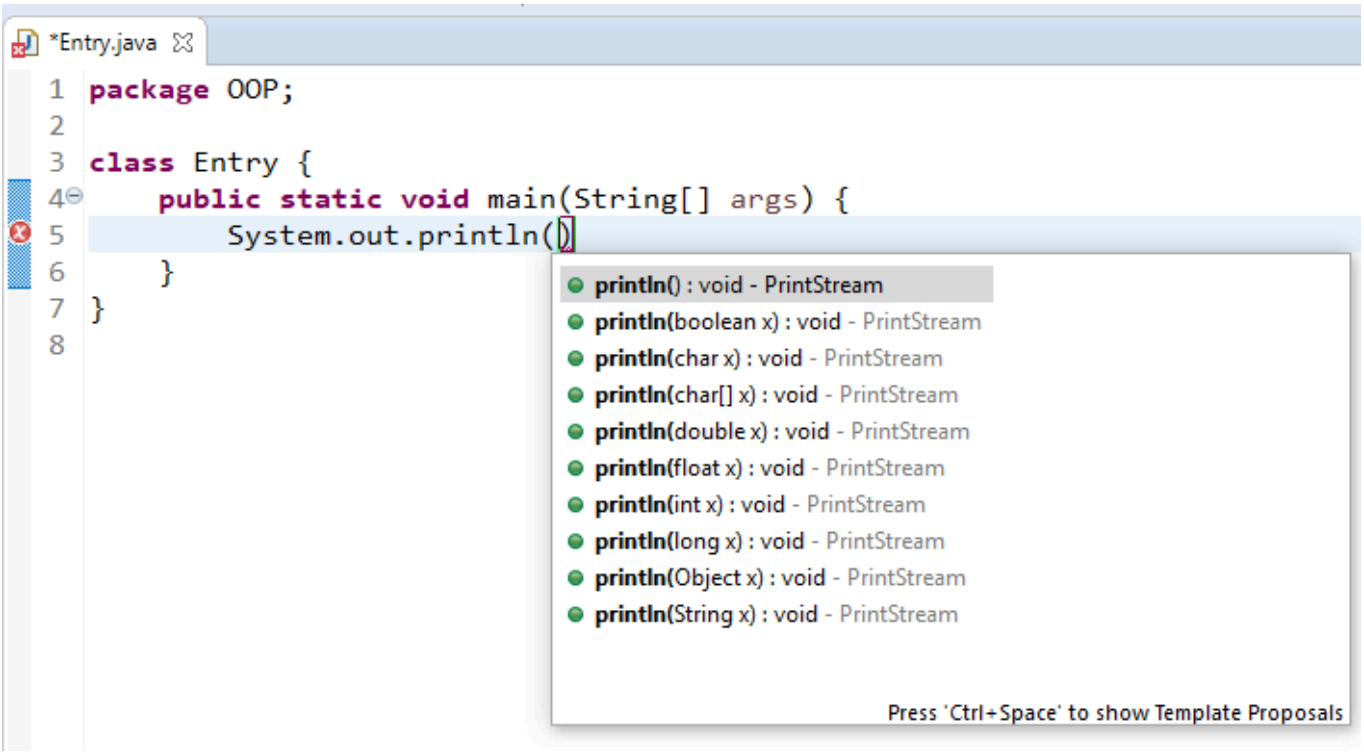
```
class Math {
    public static int max(int a, int b) {
        return (a > b) ? a : b;
    }
    public static int max(int a, int b, int c) {
        return max(max(a, b), c);
    }
}

class Entry {
    public static void main(String[] args) {
        System.out.println(Math.max(4, 5));
        System.out.println(Math.max(4, 5, 7));
    }
}
```

Kết quả khi chạy chương trình:

Lưu ý: trong các ví dụ của bài này tôi đều dùng phương thức `static` nhưng bạn hoàn toàn có thể thực hiện nạp chồng phương thức đối với các phương thức `non-static`.

Lợi ích của nạp chồng phương thức

Nạp chồng phương thức giúp bạn tránh được việc tạo ra các phương thức với tên gọi khác nhau, giúp cho code trở nên gọn gàng, dễ đọc hơn. Bạn hãy xem ví dụ về hàm `System.out.println()` mà bạn vẫn hay dùng để thấy được lợi ích của nạp chồng phương thức:



Có thể thấy nếu không có nạp chồng phương thức, thì bạn sẽ cần tới 10 cái tên cho hàm `println()` là `printlnInt()`, `printlnString()`, `printlnDouble()`, ...

Đọc tới đây bạn đã hiểu về kỹ thuật nạp chồng phương thức trong Java, hãy quay lại phần bài tập và làm thử.

Làm bài

← Bài học trước

Bài tiếp theo →

Giáo trình

Bình luận