Trong Java, `overloading` được định nghĩa là nạp chồng phương thức, có nghĩa là nếu trong một lớp có nhiều phương thức cùng tên nhưng:

- Khác nhau về số đối số truyền vào và các đối số có cùng kiểu dữ liệu.
- Có cùng số đối số truyền vào và các đối số không có cùng kiểu dữ liệu.
- Khác trình tự kiểu dữ liệu của các đối số.

Chúng ta có 2 cách để thực hiện nạp chồng phương thức đó là thay đổi số đối số truyền vào, thay đổi kiểu dữ liệu của các đối số và thay đổi trình tự của các đối số đó.

Lưu ý:

- Trong Java, chúng ta không thể thực hiện nạp chồng phương thức chỉ bằng cách thay đổi kiểu trả về của phương thức đó.
- Hàm tạo cũng có thể được nạp chồng.

Thay đổi số đối số truyền vào

PhepCongHaiSo.java

```
public class PhepCongHaiSo {
	public int add(int a, int b) {
		return a + b;
	}

	public int add(int a, int b, int c) {
		return a + b + c;
	}

}
```

TestOverriding.java

```
public class TestOverloading {
	public static void main(String[] args) {
		PhepCongHaiSo test = new PhepCongHaiSo();
		System.out.println("Tổng = " + test.add(11, 12));
		System.out.println("Tổng = " + test.add(11, 12, 13));
	}
}
```

Kết quả:

Tổng = 23
Tổng = 36

Thay đổi kiểu dữ liệu của đối số truyền vào

DisplayOverloading.java

```
public class PhepCongHaiSo {
	public int add(int a, int b) {
		return a + b;
	}

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

TestOverloading.java

```
public class TestOverloading {
	public static void main(String[] args) {
		PhepCongHaiSo test = new PhepCongHaiSo();
		System.out.println("Tổng = " + test.add(11, 12));
		System.out.println("Tổng = " + test.add(11.1f, 12.2f));
	}

}
```

Kết quả:

Tổng = 23

Tổng = 23.3

Thay đổi trình tự kiểu dữ liệu của các đối số

DisplayOverloading.java

```
public class DisplayOverloading {
	public void hienThi(int a, char b) {
		System.out.println(a + "\t" + b);
	}
```

```
	public void hienThi(char b, int a) {
		System.out.println(b + "\t" + a);
	}
}
```

TestOverloading.java

```
public class TestOverloading {
	public static void main(String[] args) {
		DisplayOverloading displayOverloading = new DisplayOverloading();
		displayOverloading.hienThi(5, 'a');
		displayOverloading.hienThi('a', 5);
	}
}
```

Kết quả:

5 a

a 5

Trường hợp chưa xác định số đối số cần truyền vào thì trong Java có một cách nạp chồng phương thức nâng cao như sau:

AdvancedOverloadMethod.java

```
package viduoverloading;

public class AdvancedOverloadMethod {
	// Dấu ... là quy tắc tạo Parameter List
	// ...x lúc này là một mảng một chiều linh động
	// tức là nó tự động co giãn được đối với các đối số truyền vào.
	public int fn4(int ...x) {
		int sum = 0;
		// duyệt qua từng phần tử trong mảng một chiều x
		for (int i : x) {
			sum += i;
		}
		return sum;
	}
}
```

TestAdvancedOverloadMethod.java

```
package viduoverloading;

public class TestAdvancedOverloadMethod {

	public static void main(String[] args) {
		AdvancedOverloadMethod demo = new AdvancedOverloadMethod();
		System.out.println(demo.fn4(1, 2, 3));
		System.out.println(demo.fn4(1, 2, 3, 4));
	}

}
```

Kết quả:

6

10

Trong lớp `TestAdvancedOverloadMethod` khởi tạo đối tượng `demo` của lớp `AdvancedOverloadMethod` và tiến hành gọi phương thức `fn4()` của lớp này với số đối số truyền vào khác nhau. Lần gọi đầu tiên, tôi truyền vào 3 số 1, 2, 3 trong khi ở lần gọi thứ hai, tôi truyền vào 4 số là 1, 2, 3, 4. Thực chất của trường hợp nạp chồng phương thức này chính là nạp chồng phương thức với số lượng các đối số khác nhau.