Buổi 4

- Nạp chồng phương thức
- Quan hệ HAS-A
- Mảng đối tượng

Nạp chồng phương thức

Nạp chồng phương thức

(method overloading)

- **Nạp chồng phương thức** trong java xảy ra nếu một lớp có nhiều phương thức có tên giống nhau nhưng khác nhau **về kiểu dữ liệu** hoặc số lượng các tham số.

- Giả sử bạn phải thực hiện tính tổng của các số đã cho với bất kỳ số lượng các đối số, nếu bạn viết phương thức a(int, int) cho 2 tham số, b(int, int, int) cho 3 tham số điều này có thể gây khó hiểu cho các lập trình viên khác về ý nghĩa của các phương thức đó vì chúng có tên khác nhau.

Lợi ích của nạp chồng phương thức

- Sử dụng nạp chồng phương thức giúp tăng khả năng đọc hiểu chương trình.

Các cách nạp chồng phương thức

Có 2 cách nạp chồng phương thức trong java

- Thay đổi số lượng các tham số
- Thay đổi kiểu dữ liệu của các tham số

Nạp chồng phương thức: thay đổi số lượng các tham số

Ví dụ: chúng ta cần tạo 2 phương thức, phương thức add() đầu tiên thực hiện việc tính tổng của 2 số, phương thức thứ hai thực hiện việc tính tổng của 3 số. Sử dụng phương thức static để gọi hàm thông qua tên class thay vì phải tạo thể hiên của lớp.

```
class Adder{
    static int add(int a,int b){return a+b;}
    static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
    public static void main(String[] args){
        System.out.println(Adder.add(11,11));
        System.out.println(Adder.add(11,11,11));
    }
}
```

Kết quả:

```
22
33
```

Nạp chồng phương thức: thay đổi kiểu dữ liệu của các tham số

Ví dụ: chúng ta sẽ tạo ra 2 phương thức có kiểu dữ liệu khác nhau. Phương thức add() đầu tiên nhận 2 đối số có kiểu giá trị là integer, phương thức thứ hai nhận 2 đối số có kiểu giá trị là double.

```
class Adder{
    static int add(int a, int b){return a+b;}
    static double add(double a, double b){return a+b;}
}
class TestOverloading2{
    public static void main(String[] args){
        System.out.println(Adder.add(11,11));
        System.out.println(Adder.add(12.3,12.6));
    }
}
```

Kết quả:

```
22
24.9
```

Tại sao không thể nạp chồng phương thức bằng cách chỉ thay đổi kiểu trả về của phương thức?

Trong java, không thể nạp chồng phương thức bằng cách chỉ thay đổi kiểu trả về của phương thức bởi vì không biết phương thức nào sẽ được gọi.

```
class Adder{
    static int add(int a,int b){return a+b;}
    static double add(int a,int b){return a+b;}
}
class TestOverloading3{
    public static void main(String[] args){
        System.out.println(Adder.add(11,11));//ambiguity
    }
}
```

Có thể nạp chồng phương thức main() không?

Có, bạn có thể nạp chồng n phương thức main. Nhưng JVM chỉ gọi phương thức main() có tham số truyền vào là một mảng String. Ví dụ:

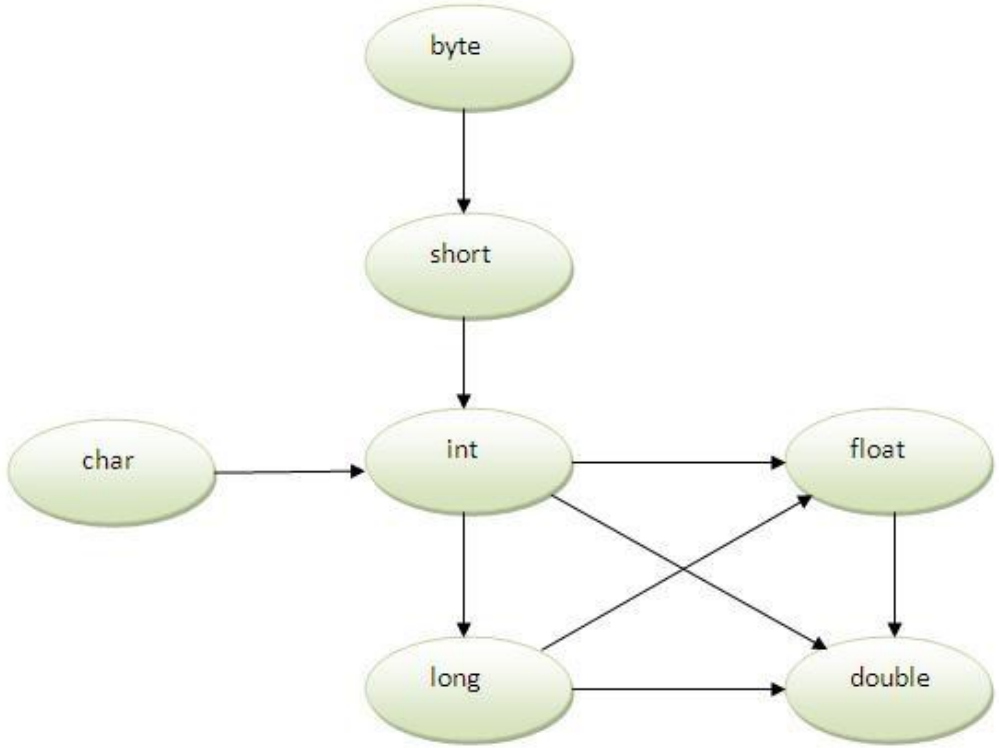
```
public class TestOverloading4 {
    public static void main(String[] args) {
        System.out.println("main with String[]");
    }

    public static void main(String args) {
        System.out.println("main with String");
    }

    public static void main() {
        System.out.println("main without args");
    }
}
```

Kết quả:

```
main with String[]
```

Nạp chồng phương thức và sự thay đổi kiểu giá trị

- Kiểu dữ liệu của đối số truyền vào được thay đổi sang kiểu dữ liệu khác (tự động ép kiểu) nếu giá trị của đối số đó không phù hợp với kiểu dữ liệu của tham số đã được đinh nghĩa:



Kiểu byte có thể được ép sang các kiểu short, int, long, float hoặc double. Kiểu dữ liệu short có thể được ép sang các kiểu int, long, float hoặc double. Kiểu dữ liệu char có thể được ép sang các kiểu int, long, float or double...

```
public class OverloadingCalculation1 {
 void sum(int a, long b) {
  System.out.println(a + b);
 }

 void sum(int a, int b, int c) {
  System.out.println(a + b + c);
 }

 public static void main(String args[]) {
  OverloadingCalculation1 obj = new OverloadingCalculation1();
  // kiểu integer tham số 2 sẽ được thay đổi thành kiểu long
  obj.sum(20, 20);
  obj.sum(20, 20, 20);
 }
}
```

Kết quả:

```
40
60
```

nếu không có kiểu đối số nào phù hợp, chuyển đổi kiểu sẽ không được thực hiện.

```java
public class OverloadingCalculation2 {
 void sum(int a, int b) {
  System.out.println("int arg method invoked");
 }

 void sum(long a, long b) {
  System.out.println("long arg method invoked");
 }

 public static void main(String args[]) {
  OverloadingCalculation2 obj = new OverloadingCalculation2();
  // phương thứ sum() có đối số int sẽ được gọi
  obj.sum(20, 20);
 }
}
```

Kết quả:

```
int arg method invoked
```

không có kiểu đối số nào phụ hợp trong phương thức và mỗi phương thức thay đổi số đối số tương tự nhau. Th này sẽ không xác định được phương thức nào được gọi.

```java
public class OverloadingCalculation3 {
 void sum(int a, long b) {
  System.out.println("a method invoked");
 }

 void sum(long a, int b) {
  System.out.println("b method invoked");
 }

 public static void main(String args[]) {
  OverloadingCalculation3 obj = new OverloadingCalculation3();
  // không xác định được phương thức nào được gọi
  obj.sum(20, 20);
 }
}
```

Kết quả:

```
Compile Time Error
```

Quan hệ HAS-A (kết tập)

Huân gáy đi

Mảng Đối tượng

Mảng đối tượng trong Java là gì?

Không giống như mảng Java truyền thống mà trong đó chứa các giá trị như:

String
Integer
Boolean
...

Mảng đối tượng trong Java chứa các **đối tượng**.

Các phần tử mảng lưu trữ các vi trí biến tham chiếu của các đối tương.

- **Cú pháp khai báo mảng đối tượng trong Java:**

```
Class obj[]= new Class[array_length];
```

Ví dụ : Đối tượng Sinh viên có 2 thuộc tính là name và age.

```java
class Student {
    String name;
    int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

Để sử dụng mảng trong Java OOP, bạn cần khởi tạo các phần tử. Ví dụ:

```java
// Declare a students array of n elements
Student[] students = new Student[n];
for (int i = 0; i < n; i++) {
    // Initialize elements in students array
    students[i] = new Student();
}
```

Mẫu mã sử dụng phương thức khởi tạo có các tham số:

```
class Student {
    String name;
    int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

public class Entry {
    public static void main(String[] args) {
        Student[] students = new Student[3];
        students[0] = new Student("Tuan", 24);
        students[1] = new Student("Cuong", 25);
        students[2] = new Student("Duc", 24);
        for (int i = 0; i < 3; i++) {
            students[i].display();
        }
    }
}
```

Kết quả:

```
Name: Tuan
Age: 24
Name: Cuong
Age: 25
Name: Duc
Age: 24
```

sử dụng phương thức khởi tạo không có tham số:

```
class Student {
    String name;
    int age;

    public Student() {

    }

    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

public class Entry {
    public static void main(String[] args) {
        Student[] students = new Student[3];
        for (int i = 0; i < 3; i++) {
            students[i] = new Student();
        }
        students[0].name = "Tuan";
        students[0].age = 24;
        students[1].name = "Cuong";
        students[1].age = 25;
        students[2].name = "Duc";
        students[2].age = 24;
        for (int i = 0; i < 3; i++) {
            students[i].display();
        }
    }
}
```

- Hết

-