Xây dựng lớp, đóng gói và phạm vi truy cập

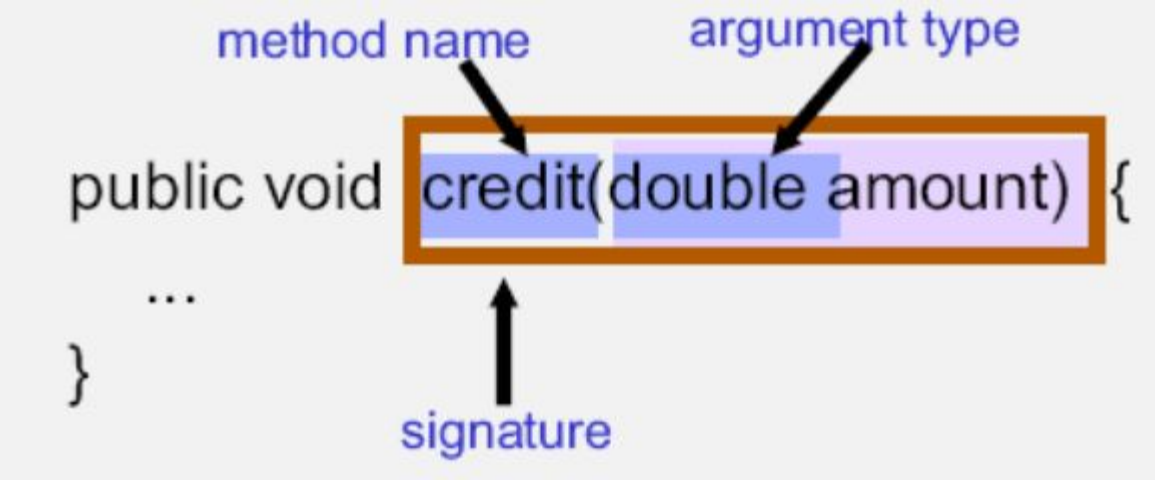
Nội dung

1. Nạp chồng
2. Đóng gói
3. Phạm vi truy cập
4. Sử dụng static và final

Nạp chồng - Overload

Chữ ký của phương thức

- tên phương thức
- số lượng các đối số truyền vào và kiểu dữ liệu



Nạp chồng - Overload

Nạp chồng hay **chồng phương thức** là cách khai báo các phương thức trong cùng một lớp có cùng tên nhưng khác chữ ký.

- Số lượng tham số khác nhau
- Kiểu dữ liệu khác nhau

Nạp chồng - Overload

Mục đích

- Tên trùng nhau để mô tả bản chất công việc
- Thuận tiên cho việc lập trình vì không phải nhớ quá nhiều tên mà chỉ cần lựa chọn tham số phù hợp.

Nạp chồng - Overload

Ví du

```
class MyDate {
    int month;
    public void setMonth(int month) {...}
    public void setMonth(String month) {...}
}
```

Nạp chồng - Overload

Ví du trong Java

```
System.out.println(int x);
System.out.println(boolean x);
System.out.println(char x);
System.out.println(double x);
System.out.println(String x);
...
```

Đóng gói

Đóng gói - package

- Một tập hợp các Lớp - class có liên quan đến nhau hoặc có cùng chức năng được gom lại trong cùng một namespace được gọi là 1 package

Đóng gói - package

Đặc điểm của 1 Java package

- 1 package có thể chứa nhiều package con
- Trong cùng 1 package không thể có 2 member cùng tên
- Tên package được viết thường toàn bộ
- Các package tiêu chuẩn của java được bắt đầu bằng từ khoá ***java*** hoặc ***javax***
- Các package con sẽ được truy cập đến bằng cách sử dụng " **.** "

Đóng gói - package

Ví dụ các package trong java

- java.lang
- java.util
- java.io

Đóng gói - package

Sử dung package

- Để sử dụng package trong java chúng ta sử dụng từ khoá **`"import"`**
- Có 2 loại **`import`**
  - import chính xác class / interface cần dùng
  - import toàn bộ gói

Đóng gói - package

Ví dụ

```
import java.util.ArrayList
import java.util.Scanner
import java.util.*
```

Phạm vi truy cập

Phạm vi truy cập

- Phạm vi truy cập (access modifiers) trong Java được dùng cho Class và các Thành phần trong Class như thuộc tính và phương thức.
- Phạm vi truy cập nhằm xác định khả năng truy cập, sử dụng Class từ các Class hoặc Interface khác.

Phạm vi truy cập

- Trong Java hỗ trợ các phạm vi truy cập như sau:
    - `public`
    - `private`
    - `protected`
    - `default`

Phạm vi truy cập

Sử dụng **`public với Class`**

- Trong trường hợp này thì mọi Class hay Interface khác từ bên ngoài (cùng package hoặc khác package) đều có thể truy cập và khởi tạo Class này

Sử dụng **`defaut Class`**

- Khác với public khi sử dụng default Class, chỉ các class trong cùng package mới có thể truy cập và sử dụng Class này

Phạm vi truy cập

Ví dụ

```
public class Person() { //public class
    ...
}

class Student() { // default class
    ...
}
```

Phạm vi truy cập

Sử dụng **`public với Member Class`**

- Tất cả các Class bên ngoài (cùng hoặc khác package) đều có quyền truy cập trực tiếp đến các thành viên này của Class

Sử dụng **`private Member Class`**

- Khi sử dụng private cho các Member Class, không có Class nào có thể truy cập sử dụng được các thành viên này.

Phạm vi truy cập

Sử dụng **`protected với Member Class`**

- Chỉ cho phép các Class là con của Class này hoặc chính bản thân Class được quyền truy cập, ngoài ra đều không được quyền truy cập.

Phạm vi truy cập

Ví dụ

```
public class Person() { //public class
    private int age;
    private String name;
    public int getAge() { return age; }
    protected String getName() { return name; }
}
```

Sử dụng static và final

Sử dụng static và final

Từ khoá **`static`** được dùng để khai báo các thành viên Tĩnh trong Class

- chỉ khởi tạo duy nhất 1 lần
- địa chỉ con trỏ cố định
- những lần gọi sau sẽ dùng lại mà không cần khai báo mới

Sử dụng static và final

Cú pháp khai báo

```
class Student {
    public static String id; //thuộc tính
    public static void printInfo() { //phương thức
        ...
    }
}
```

Sử dụng static và final

Cú pháp sử dụng

```
class Main {
    public static void main(String[] args){
        Student.id = “ST001”;
        Student.printInfo();
    }
}
```

Sử dụng static và final

Từ khoá **`final`** được dùng để khai báo các thành viên Hằng trong Class

- áp dụng cho cả thuộc tính và phương thức
- không thể thay đổi giá trị trong quá trình sử dụng

Sử dụng static và final

Cú pháp khai báo

```
[phạm_vi] final <kiểu> tên_hằng = giá_trị;
```

Ví dụ

```
public final int PI = 3.14;
final String VERSION = "1.1";
```