HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
Posts and Telecommunications Institute of Technology

RIPT

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Nguyễn Việt Dũng

dungnv1@ptit.edu.vn

XÂY DỰNG LỚP VÀ ĐỐI TƯỢNG

XÂY DỰNG LỚP

Khai báo lớp (Class)

```
[package package-name;]
[access-modifier] class class-name {
   // class body
   access-modifier type instance-variable-1; …
   access-modifier static type class-variable-2; ...
   access-modifier type method-name-1(parameter-list)
   {…} …
}
```

Phạm vi truy cập (Access Modifier)

- public: Hiển thị và tự do truy cập từ bên ngoài
- private: Chỉ hiển thị và sử dụng trong class
- protected: Chỉ hiển thị và sử dụng trong cùng package và lớp con (kế thừa)
- default (không khai báo): Chỉ hiển thị và sử dụng trong cùng package

Phạm vi truy cập (Non - Access Modifier)

final: Thuộc tính và phương thức không thể bị ghi đè

static: Thuộc tính và phương thức tĩnh của lớp chứ không phải của đối tượng

abstract: Chỉ sử dụng cho phương thức trong lớp trừu tượng. Không có code thực thi bên trong mà sẽ được cung cấp ở lớp con

transient: Thuộc tính và phương thức sẽ bị bỏ qua khi serializing đối tượng

synchronized: Phương thức chỉ có thể sử dụng trong 1 luồng 1 lúc (tránh truy cập đồng thời)

volatile: Giá trị của thuộc tính không được cache trong luồng mà luôn đọc từ "main memory"

Khai báo biến trong lớp (variable)

- Instance variables (member variables): khai báo trong lớp, ngoài bất kỳ các phương thức nào
- Class variables: khai báo trong lớp, ngoài bất kỳ các phương thức nào với từ khóa static
- Local variables: khai báo và sử dụng trong phương thức, đoạn code, constructor

```
[access-modifier] class class-name {
    // class body
    access-modifier type instance-variable-1; …
    access-modifier static type class-variable-2; ...
    }
```

Phạm vi truy cập

Khai báo 2 lớp dưới đây

```
class Account {
        String owner;
        long balance; …
}

class PrivateAccount {
        private String owner;
        private long balance; …
}
```

Khởi tạo biến trong lớp (variable)

```
datatype var1 = new datatype();
```

Khai báo biến var1 là một đối tượng của lớp

```
Account acct1;  // acct1 represents nothing
```

Sau khi gán giá trị, tham chiếu của var1 sẽ chứa địa chỉ của đối tượng lớp tạo trong bộ nhớ.

```
acct1 = new Account(); // acc1 refers to an Account1 object
```

Khởi tạo biến trong lớp (variable)

Mỗi đối tượng có 1 vùng bộ nhớ riêng biệt

Thay đổi giá trị biến trong đối tượng này không làm ảnh hưởng đến giá trị biến tương tự ở đối tượng khác.

```
Account acc1 = new Account(“Nguyen Viet Dung”, 10000000);
Account acc2 = acc1;
acc2.setOwner(“Tung Lam”);
acc2.setBalance(20000000);
acc1.getBalance();
```

Class (static) variables

```
[access-modifier] static data-type member-variable-name;
```

- Class variable: khai báo với từ khóa static
- Chứa chung dữ liệu với tất cả đối tượng trong lớp (thay đổi ở 1 đối tượng cũng sẽ thay đổi ở mọi đối tượng khác cùng lớp)

Class (static) variables

```
class Account {
   // Member variable
   String name; // Account name
   long balance; // Balance
   public static float interest; // Deposit rate
}

Account.interest = 0.05f;

acc01.interest = 0.07f;
```

Khai báo phương thức

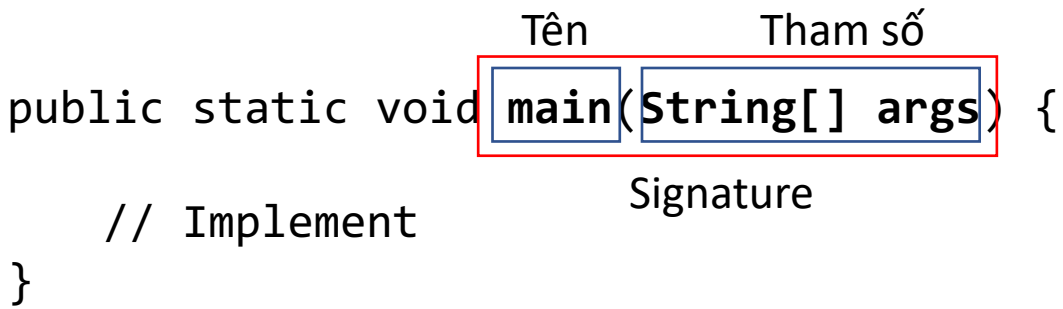
```
// Declaration
[access-modifier] return-type name(parameter-list) {
    // Implement
    return return-value;
}
```

1. Declaration
    - Tên phương thức
    - Kiểu dữ liệu trả về
    - Tham số đầu vào
2. Implement
    - Tập câu lệnh thực hiện khi gọi phương thức
    - Có thể thay đổi hoặc cập nhật trạng thái của đối tượng, biến

Chữ ký của phương thức (Signature)

Cấu thành bởi

- Tên phương thức
- Tập tham số đầu vào của phương thức theo đúng số lượng, kiểu, thứ tự



Chữ ký của phương thức (Signature)

```
public static void main(String[] args) {
    System.out.println(”Main string[] args");}
}

public static void main(String args) {
    System.out.println("Call main(String args) method: " + args);
}

public static void main(String args1, String args2) {
    System.out.println("Call main(String args1, String args2)
method: " + args1 + args2);
}
```

Chữ ký của phương thức (Signature)

```
public static void main(String args) {    Duplicate method main(String) in type Main
    System.out.println("Call main(String args) method: " + args);

}

public static int main(String args) {    Duplicate method main(String) in type Main
    System.out.println("Call main(String args) method: " + args);

}
```

Static methods

```
// Declaration
static return-type name(parameter-list) {
    // Implement
}
```

Lưu ý

– Có thể gọi theo 2 cách

- Class name.method_name(argument);
- Object name.method_name(argument);

```
car01.showVehicle();    The static method showVehicle() from the type Vehicle should be accessed in a static way
Vehicle.showVehicle();
```

– Chỉ có thể truy cập các biến static

```
public static void showVehicle() {
    System.out.println(TYPE);
    System.out.println(this.plateNumber);    Cannot use this in a static context
}
```

Phương thức với tham số có số lượng động

```
public class VariableLengthArgumentUsage {
    public static double average( double... numbers ) {
        double total = 0.0;
        for ( double d : numbers )
            total += d;
        return total / numbers.length;
    }

    public static void main(String[] args) {
        double d[] = {10.0, 20.0, 30.0, 40.0, 50.0};
        System.out.println(”Average value of this array is");
        System.out.println(" %.1f\n", average(d[0], d[1], d[2], d[3], d[4]));
    }
}
```

XÂY DỰNG ĐỐI TƯỢNG

Khởi tạo dữ liệu

Cần khởi tạo đối tượng trước khi sử dụng.

Trong đó:

- Kiểu dữ liệu nguyên thủy (Primitive data type): Khởi tạo bằng phép gán

- Đối tượng (theo lớp): Khởi tạo thông qua **constructor** của lớp

Constructor

- Người dùng định nghĩa nếu không Java tự quy định mặc định với tên là tên lớp, không có tham số đầu vào và giá trị mặc định của các thuộc tính theo kiểu dữ liệu
- Có thể có hoặc không có tham số đầu vào

```
public class Vehicle {
    private String plateNumber;
    private int numberOfWheels;
    private int numberOfPassengers;

    public Vehicle(int numberOfWheels, int numberOfPassengers) {
        this.numberOfWheels = numberOfWheels;
        this.numberOfPassengers = numberOfPassengers;
    }
}
```

Khởi tạo đối tượng

```
// Declaration
datatype instance-variable;
// Initialization
instance-variable = new datatype();
```

Hoặc

```
datatype instance-variable = new datatype();
```

Sử dụng đối tượng

- Truy cập và lấy trạng thái của một thuộc tính của đối tượng
- Gọi và thực thi một phương thức của đối tượng

Sử dụng dấu “.”

```
car01.publicName;
car01.showInfo();
bus01.showInfo();
```

Từ khóa this

Dùng trong một phương thức hoặc constructor để tham chiếu đến đối tượng đang xét, giúp phân biệt thuộc tính của đối tượng với tham số truyền vào

```
public class Vehicle {
    private String plateNumber;
    private int numberOfWheels;
    private int numberOfPassengers;

    public Vehicle(int numberOfWheels, int numberOfPassengers) {
        this.numberOfWheels = numberOfWheels;
        this.numberOfPassengers = numberOfPassengers;
    }
}
```

Truyền tham số

- Truyền theo tham chiếu

- Truyền theo giá trị

Bài tập

Bài toán: Quản lý học sinh và lớp học

Xây dựng một hệ thống đơn giản quản lý học sinh và lớp học. Mỗi học sinh sẽ thuộc về một lớp học cụ thể, và lớp học có danh sách các học sinh của mình.

Yêu cầu:

1. Tạo lớp Student (Học sinh) với các thuộc tính cơ bản như tên, tuổi và phương thức hiển thị thông tin học sinh.
2. Tạo lớp Classroom (Lớp học) với thuộc tính tên lớp và một danh sách các học sinh.
3. Lớp học có thể thêm học sinh vào danh sách và hiển thị thông tin của tất cả học sinh trong lớp.