Chương 2

Lớp và Đối tượng

Giáo viên : ThS. Trần Văn Thọ
Đơn vị : Bộ môn KTHT & MMT

1. Lớp và đối tượng

- Lớp là một kiểu dữ liệu trừu tượng
  - Phản ánh một khuôn mẫu của thực thể dữ liệu hay mô hình dữ liệu
  - Gồm 2 thành phần:
    - Thuộc tính: các thành phần mang dữ liệu
    - Phương thức: các hàm tương tác với các thuộc tính
  - Trong Java, mọi hàm và biến dữ liệu đều phải nằm trong khai báo lớp (không có khái niệm hàm tự do hay biến toàn cục)
- Đối tượng là một thể hiện của lớp:
  - Đại diện cho thực thể mang dữ liệu thực sự
  - Một đối tượng là một biến có kiểu là lớp
- Một lớp có thể có bao nhiêu đối tượng tùy ý.

1. Lớp và đối tượng

Định nghĩa lớp:
Cú pháp:
class <tên lớp> {
<các thuộc tính>
<các phương thức>
};

```
class Employee{
    int id;
    String name;
    float salary;
    void insert(int i, String n, float s) {
        id=i;
        name=n;
        salary=s;
    }
    void display(){
            System.out.println(id+" "+name+" "+salary);
     }
}
```

1. Lớp và đối tượng

Khai báo biến thành phần trong lớp:

```
<kiểu> <tên biến>;
```

Khai báo hàm thành phần trong lớp

```
<kiểu trả về> <tên hàm>(<danh sách tham số>){}
```

- Tham số trong hàm thành phần được truyền mặc định kiểu tham trị, nhưng nếu tham số có kiểu đối tượng thì giá trị tham số sẽ là tham chiếu của đối tượng truyền vào.

1. Lớp và đối tượng

Khai báo đối tượng:

Cú pháp:

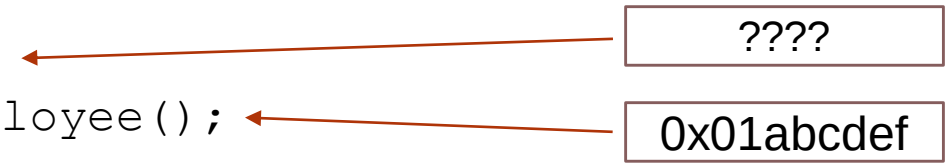
```
<tên lớp> <tên đối tượng>;
```

Sử dụng từ khóa new để khởi tạo đối tượng:

```
<tên đối tượng> =new <tên lớp>();
```

VD:

```
Employee me;
me = new Employee();
```



Đối tượng trong Java thực chất là các tham chiếu

-> địa chỉ trỏ đến vùng nhớ được cấp phát động bằng **new**

1. Lớp và đối tượng

Gọi phương thức và truy cập thuộc tính từ đối tượng:
Cú pháp:

```
<tên đối tượng>.<phương thức>(<tham số>);
<tên đối tượng>.<tên thuộc tính>;
```

```
VD:
Employee me;
me = new Employee();
me.insert(1,”An”,10000);
me.display();
```

output

```
1 An 10000
```

2. Phạm vi lớp

2.1 Gói (package) trong Java

- Nhóm các lớp, giao diện (interfaces) và các gói khác
- Lợi ích:
  - Quản lý mã nguồn hiệu quả hơn
  - Cung cấp cơ chế quản lý truy cập
  - Tránh xung đột về tên
- Các loại gói trong Java:
  - Gói có sẵn trong java (built-in package)
  - Gói do người lập trình định nghĩa

2. Phạm vi lớp

2.1 Gói (package) trong Java

- Khai báo gói trong file .java:

```
package <tên gói>;
```

- Truy cập một gói trong phạm vi gói khác:
  - Mở toàn bộ phạm vi một gói:
    - VD : `import mypackage.*;`
  - Mở một lớp trong gói:
    - VD: `import mypackage.Myclass;`
  - Chỉ dẫn tường minh:
    - VD: `mypackage.Myclass myclass = new mypackage.Myclass();`

2. Phạm vi lớp

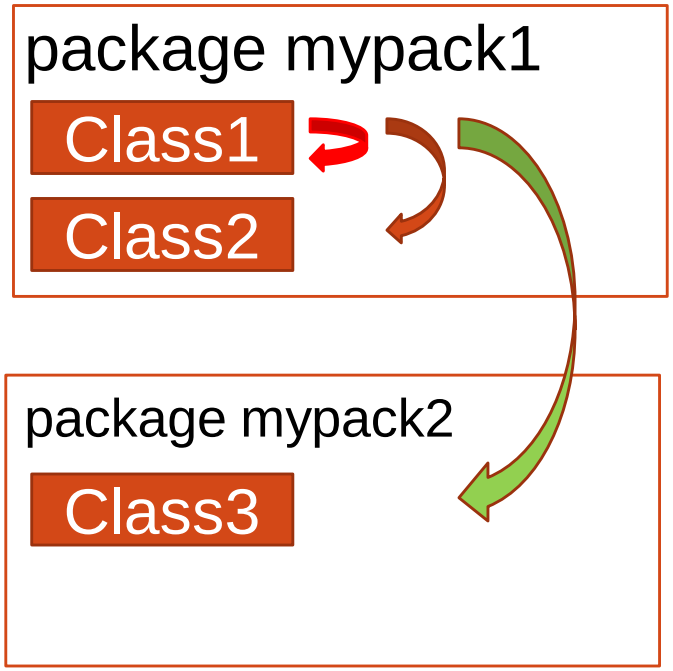
2.2 Phạm vi truy cập các thành phần của lớp:

- Việc gọi hàm và thuộc tính của đối tượng được giới hạn bởi phạm vi truy cập -> tính chất đóng gói của lập trình hướng đối tượng.

- Các loại phạm vi:
  - Bên trong lớp : gọi hàm hay thuộc tính của lớp từ một hàm bên trong lớp đó
  - Bên trong gói: gọi hàm hay thuộc tính của lớp từ một hàm nằm bên trong lớp khác trong cùng một gói
  - Bên ngoài gói: gọi hàm hay thuộc tính của lớp từ một hàm nằm bên trong lớp thuộc gói khác

2. Phạm vi lớp

2.2 Phạm vi truy cập các thành phần của lớp:




```
Class1{
        void insideFunction(){...}
        void access(){
                insideFunction();
                Class2 c2=new Class2();
                c2….
                Class3 c3=new Class3();
                c3…
        }
}
```

2. Phạm vi lớp

2.2 Phạm vi truy cập các thành phần của lớp:

Java cung cấp 4 mức truy cập cho mỗi thành phần của lớp

- **private**:

Các thành phần sau từ khóa này được bảo vệ và không thể truy cập từ bên ngoài, chỉ có thể truy cập bởi các hàm thành phần của lớp.

- **default:**

Phạm vi mặc định. Nếu không sử dụng từ khóa phạm vi truy cập nào thì phạm vi truy cập là chỉ bên trong gói.

- **protected** :

Các thành phần sau từ khóa này có phạm vi truy cập chỉ bên trong gói và ngoài gói thông qua các lớp con.

- **public :**

Các thành phần sau từ khóa này có phạm vi truy cập bất kỳ.

3. Chồng hàm thành phần

- Cho phép nhiều hàm trong một lớp có thể trùng tên nhau nhưng phải khác danh sách tham số (số lượng, kiểu, thứ tự tham số).
  - Kiểu trả về có thể khác nhau
- Khi gọi hàm thành phần từ đối tượng, dựa vào danh sách tham số mà trình biên dịch sẽ chọn hàm thành phần tương ứng để thực hiện
- VD:

```
public void println(int i)
public void println(float f)
public void println(String s)
```

4. Hàm thiết lập

- Là một hàm thành phần đặc biệt dùng để khởi tạo giá trị trong lớp và thực hiện các công việc chuẩn bị khác.
- Là hàm không thể thiếu, ngay cả khi người lập trình không định nghĩa thì chương trình cũng sẽ tự tạo một hàm thiết lập ngầm định cho lớp.
- Tự động được gọi khi khởi tạo đối tượng bằng từ khóa **new**.
- Có thể có tham số hoặc không có tham số
  - Tham số được truyền khi khởi tạo bằng từ khóa **new**
- Có thể định nghĩa chồng hàm với hàm thiết lập
- Tên hàm thiết lập phải **trùng tên lớp** và **không** có kiểu trả về.

4. Hàm thiết lập

VD:

```
public class Employee {
 private String name;
 private int salary;
 public Employee(String n, int s)
{
       name = n;
       salary = s;
}
public Employee(String n) {
       name=n;
       salary=0;
}
public Employee() {
       name= "unknown";
}
}
```

```
public static void
main(String[] args) {
Employee e1=new
Employee("An",10000);
Employee e2=new
Employee("Binh");
Employee e3=new Employee();

}
```

4. Hàm thiết lập

Hàm thiết lập mặc định:

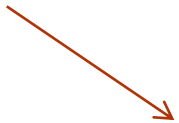
- Là hàm thiết lập được tự động sinh ra nếu người dùng không định nghĩa hàm thiết lập nào.
- Là hàm thiết lập không có tham số
- Cho phép tạo đối tượng với từ khóa new mà không truyền tham số

VD: Employee e=new Employee();

5. Các thành phần tĩnh

Xét cú pháp gọi phương thức sau:

```
<đối tượng>.<tên phương thức>(tham số);
```



phải tạo ra đối tượng trước

Trong nhiều trường hợp, điều này là không cần thiết

Cần có các thành phần chỉ gắn với lớp, không cần gắn với các đối tượng cụ thể

VD: giả sử có lớp Math cung cấp các hàm toán học, khi sử dụng hàm sqrt ta phải gọi như sau:

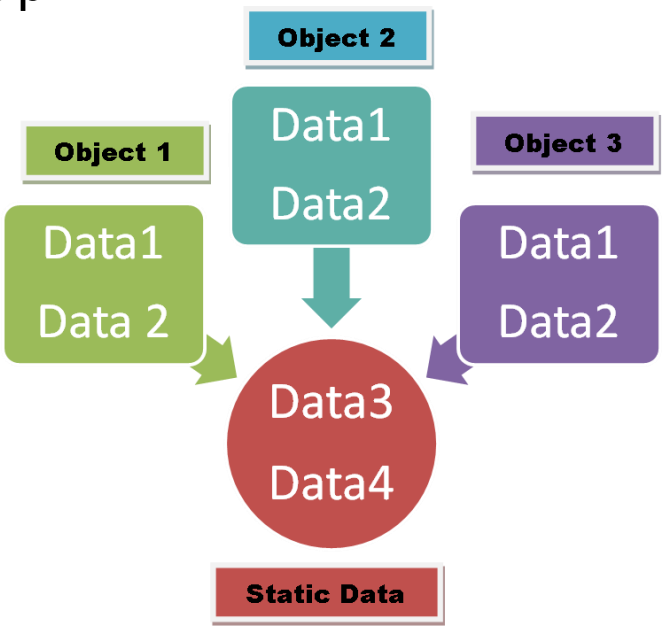
```
Math m =new Math();
float y=m.sqrt(x);
```

Chỉ cần hàm thành phần mà không cần thuộc tính của m -> không cần thiết phải khai báo và khởi tao m

5. Các thành phần tĩnh

Thành phần tĩnh:

- Thành phần được sử dụng chung cho tất cả các đối tượng thuộc cùng lớp
- Chỉ có 1 thể hiện duy nhất được chia sẻ bởi tất cả các đối tượng thuộc cùng lớp



5. Các thành phần tĩnh

Thành phần dữ liệu tĩnh:

- Chỉ có 1 thể hiện duy nhất: dữ liệu dùng chung cho mọi đối tượng của lớp
- Có thể có các thuộc tính truy cập khác nhau
- Cú pháp khai báo: đặt từ khóa static trước thành phần dữ liệu. Có thể khởi tạo giá trị cho dữ liệu tĩnh ngay khi khai báo.
  - VD: `static int count=0;`

5. Các thành phần tĩnh

Hàm thành phần tĩnh:

- Là các hàm chỉ dùng để đọc và thay đổi giá trị của thành phần dữ liệu tĩnh.
  - Không làm việc với từng đối tượng cụ thể
  - Không sử dụng các thành phần dữ liệu khác của lớp
- Có thể khai báo bên trong hoặc bên ngoài lớp
  - Cú pháp: thêm từ khóa static trước khai báo hàm thành phần.
  - VD: static int getcount();
- Gọi hàm tĩnh:
  - Gọi qua tên lớp :
    - <tên lớp>.<tên hàm thành phần tĩnh>(ds tham số);

5. Các thành phần tĩnh

```
class Counter {
    static int count = 0;
    Counter() {
        count++;
    }
    public static int
getCount(){
        return count;
    }
}
```

```
public static void
main(String[] args) {
Counter c1 = new
Counter();

System.out.println(
Counter.getCount());
Counter c2 = new
Counter();

System.out.println(
Counter.getCount());

    }
```

5. Các thành phần tĩnh

Nguyên tắc:

- Các hàm thành phần đều có thể truy cập vào thành phần tĩnh.
- Giá trị của thành phần tĩnh thay đổi sẽ tác động lên tất cả các đối tượng của lớp.
- Hàm thành phần tĩnh có thể được gọi mà không cần qua đối tượng

6. Từ khóa this

- Từ khóa **this** là biến tham chiếu đến chính đối tượng hiện tại.
- Sử dụng:
  - Đại diện cho đối tượng của lớp hiện tại ở trong các phương thức của lớp.
  - Có thể dùng để gọi các hàm và thuộc tính ở trong các phương thức của lớp
  - this() gọi đến hàm thiết lập của lớp một cách tường minh bên trong lớp
  - Có thể dùng để truyền như một tham số
  - Trả về đối tượng hiện tại trong các phương thức của lớp

6. Từ khóa this

VD:

```
class Student{
    int id;
    String name;
    public Student(int id,String
name){
        this.id=id;
        this.name=name;
    }
    public Student(String name){
        this(0,name);
    }
    public Student(Student s){
        this.id=s.id;
        this.name=s.name;
    }
```

```
 public Student getInstance(){
        return this;
    }
 public Student makeCopy(){
        Student s=new
Student(this);
        return s;
}
    public void display(){

System.out.println(this.id+"
"+this.name);
    }
}
```

6. Từ khóa this

VD: (tiếp)

```
public static void main(String[] args) {
        // TODO code application logic here
        Student s=new Student("An");
        s.getInstance().display();
        s.makeCopy().display();

    }
```

BÀI TẬP

1. Xây dựng lớp SinhVien

- Thuộc tính:
  - Họ tên : chuỗi ký tự
  - Lớp:chuỗi ký tự
  - Điểm trung bình :float
  - Điểm môn học:mảng float
  - Số môn học: integer;
- Phương thức:
  - Hàm khởi tạo không tham số
  - Hàm khởi tạo vơi 3 tham số: họ tên, lớp, số môn học
  - Hàm thiết lập giá trị cho điểm môn học
  - Hàm lấy ra điểm môn thứ k của sinh viên
  - Hàm tính và trả về điểm trung bình
  - Hàm so sánh 2 sinh viên theo điểm trung bình

BÀI TẬP

1. Xây dựng lớp Point3D bao gồm các thành phần sau:

- Thuộc tính:
  - Tọa độ x:integer
  - Tọa độ y: integer;
  - Tọa độ z: integer;
- Phương thức:
  - Hàm khởi tạo vơi 3 tham số: x,y,z
  - Hàm khởi tạo không tham số, thiết lập x=0,y=0,z=0
  - Hàm move với 3 tham số x,y,z
  - Hàm display hiển thị x,y,z ra màn hình

BÀI TẬP

1. Xây dựng lớp Car bao gồm các thành phần sau:

- Thuộc tính:
  - Màu sắc : chuỗi ký tự
  - Tốc độ: integer
  - Giá tiền: float
  - Tọa độ x:integer
  - Tọa độ y: integer;
- Phương thức:
  - Hàm khởi tạo vơi 3 tham số: tốc độ, x,y
  - Hàm thiết lập giá trị cho thuộc tính màu và giá tiền
  - Hàm tính khoảng cách giữa 2 đối tượng car (dựa trên tọa độ x,y)

BÀI TẬP

1. Xây dựng lớp Matrix :
   - thuộc tính:
     - Số hàng
     - Số cột
     - Ma trận: Mảng 2 chiều float

- Phương thức:
  - Hàm nhập dữ liệu cho ma trận
  - Hàm in ma trận ra màn hình
  - Chồng toán tử nhân 2 ma trận