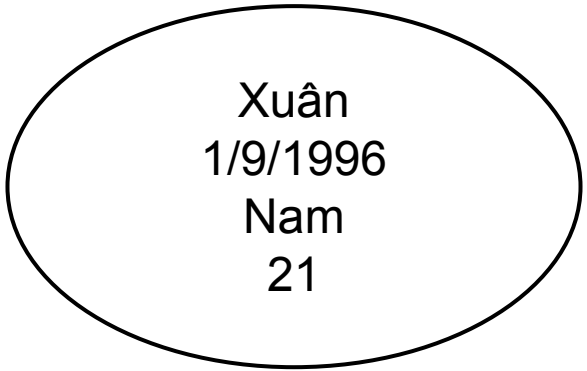
Lớp và đối tượng

Lập trình hướng đối tượng

Đối tượng là gì?

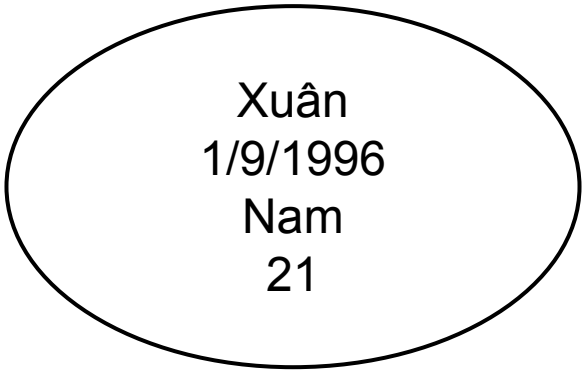
Lập trình hướng đối tượng

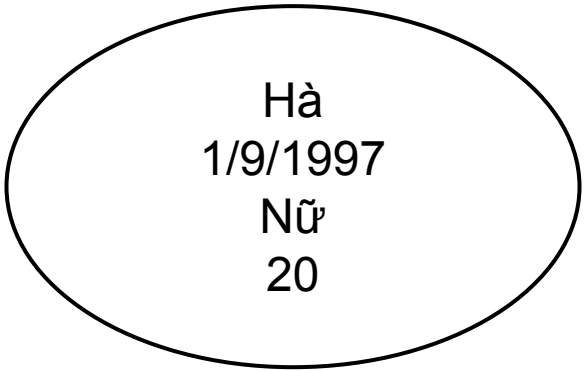
Đối tượng là gì?



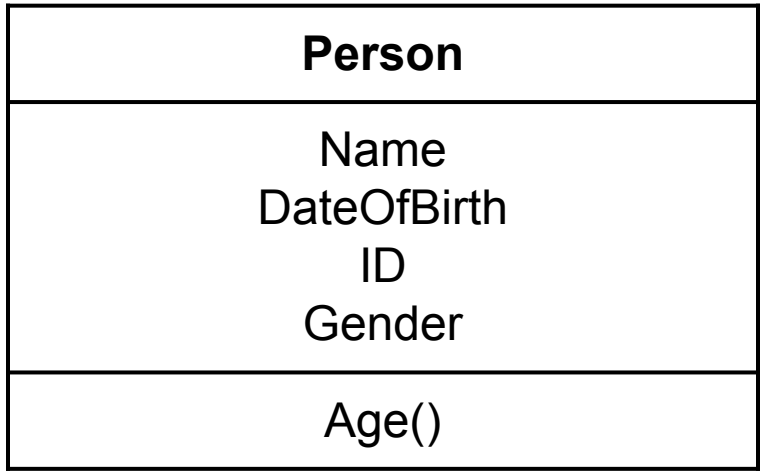
Lập trình hướng đối tượng

Đối tượng là gì?

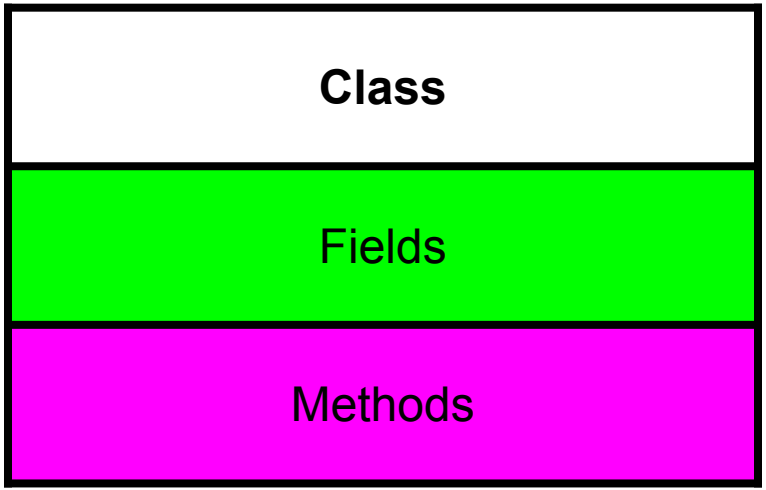




Lập trình hướng đối tượng



Lập trình hướng đối tượng



Khai báo lớp trong java

```
class Person {
        String name;
        ……..
        int Age() {
                return calAge();
        }
}
```

Khai báo đối tượng

```
Person person = new Person();
```

Constructors

```
class ClassName {

        ClassName() {

        }


        ClassName([arguments]) {

        }

}
```

Constructors

```
class Person {
        String name;
        ……..
        public Person(String _name) {
                name = _name;
        }
}
```

Constructors

- Tên constructor trùng với tên của lớp (class)
- Không có kiểu dữ liệu trả về
- Thường dùng để khởi tạo cho các trường dữ liệu (fields)
- Tất cả các lớp đều có ít nhất một constructor
  - Nếu không viết, mặc định là
  - ClassName() {}

Khai báo đối tượng

```
ClassName instance1 = new ClassName();

ClassName instance2 = new ClassName([arguments]);
```

Truy cập trường dữ liệu

- Cú pháp: Object.fieldName
- Ví dụ:

```
Person person = new Person(‘Xuân’);

System.out.println(person.name);
```

Gọi một phương thức

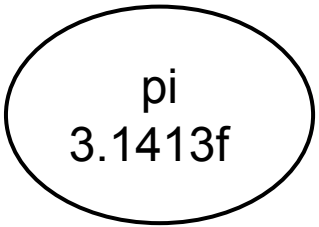
- Cú pháp: Object.methodName([params]);
- Ví dụ:

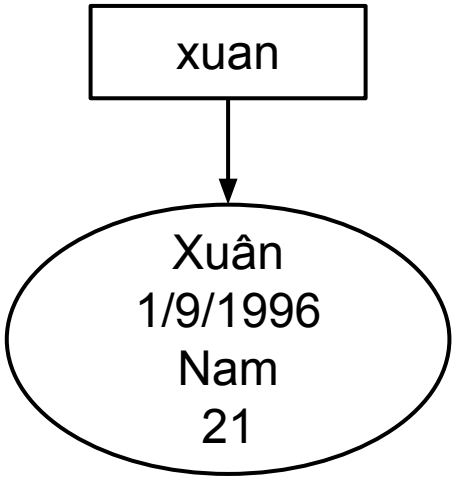
```
Person person = new Person(‘Xuân’);

System.out.println(person.Age());
```

Tham chiếu và giá trị

Ví dụ:

```
float pi = 3.1413f;

Person xuan = new person('Xuân');
```

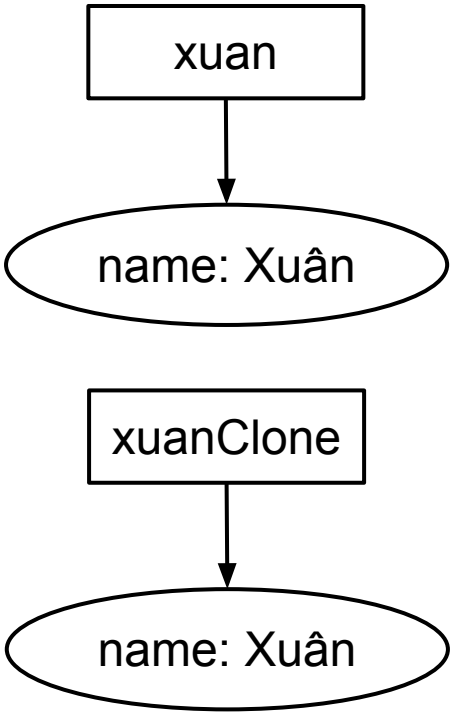




Tham chiếu và giá trị

Ví dụ:

```
Person xuan = new person(‘Xuân’);

Person xuanClone = new person(‘Xuân’);

if (xuan == xuanClone) {

    System.out.println(‘xuan = xuanClone’);

}
```



Từ khóa ‘this’

```
class Person {
    String name;
    ……..
    public Person(String _name) {
        name = _name;
    }
}
```

```
class Person {
    String name;
    ……..
    public Person(String name) {
        name = name; // ???
    }
}
```

Từ khóa ‘this’

```
class Person {
    String name;
    ……..
    public Person(String _name) {
        name = _name;
    }
}
```

```
class Person {
    String name;
    ……..
    public Person(String name) {
        this.name = name;
    }
}
```

Packages

- Mỗi lớp phải thuộc một gói
- Các lớp trong cùng một gói thì có mục đích gần tương tự nhau
- Để sử dụng các lớp ở gói khác cần phải thực hiện khai báo import
- Gói giống như các thư mục

Packages

Khai báo:

```
package path.to.package.person;

class Person {

}
```

Sử dụng:

```
import path.to.package.person.Person;

import path.to.package.person.*;
```

Packages

```
package path.to.package.person;


import path.to.package.util;


class Person {

}
```

Modifier

- Access modifiers
  - public
  - protected
  - private
- Non-access modifiers
  - static
  - final
  - abstract
  - volatile
  - synchronized

Access modifiers

Đối với lớp (class):

public - truy cập được từ mọi gói

Không có modifier - chỉ truy cập được từ gói chứa lớp

Access modifiers

Đối với các thành phần trong một lớp (class member):

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	**N**
no modifier	Y	Y	**N**	**N**
private	Y	**N**	**N**	**N**

Access modifiers

Ví dụ:

```
public class Person {
    private String name;
    public setName(String _name) {
        name = _name;
    }
}
```

Non-access modifiers (static, final)

- static
  - Dùng cho trường (fields) hoặc phương thức (method)
  - Không dùng được cho biến cục bộ
  - Nếu là biến dùng chung cho các đối tượng khai báo từ cùng một lớp.
  - Nếu là phương thức gọi mà không cần khai báo đối tượng. Tuy nhiên chỉ truy cập được các biến hoặc phương thức được khai báo là static
- Final
  - Sử dụng cho các biến (biến cục bộ hoặc trường) hoặc phương thức
  - Chỉ cho phép biến được khởi tạo một lần (gán một giá trị duy nhất)
  - Phương thức sẽ không thể được tái khai báo bởi lớp con (overridden)

Non-access modifiers (static, final)

Khai báo hằng trong java: sử dụng kết hợp static và final

```
public class HttpError {

    public static final int FILE_NOT_FOUND = 404;

}
```