

Chương 2. Lớp và đối tượng

Mục đích



- Giới thiệu các khái niệm cơ bản của lập trình hướng đối tượng.
- Trang bị các kỹ năng xây dựng lớp và các thành phần của lớp.
- Sau khi kết thúc chương có thể đặc tả và giải quyết các bài toán dựa trên hướng đối tượng.

Chương 2. Lớp và đối tượng

Nội dung



- Đối tượng
- Lớp
- Hàm thiết lập và hàm huỷ bỏ
- Các thành phần tĩnh
- Bài tập + Kiểm tra

Chương 2. Lớp và đối tượng

Đối tượng (object)



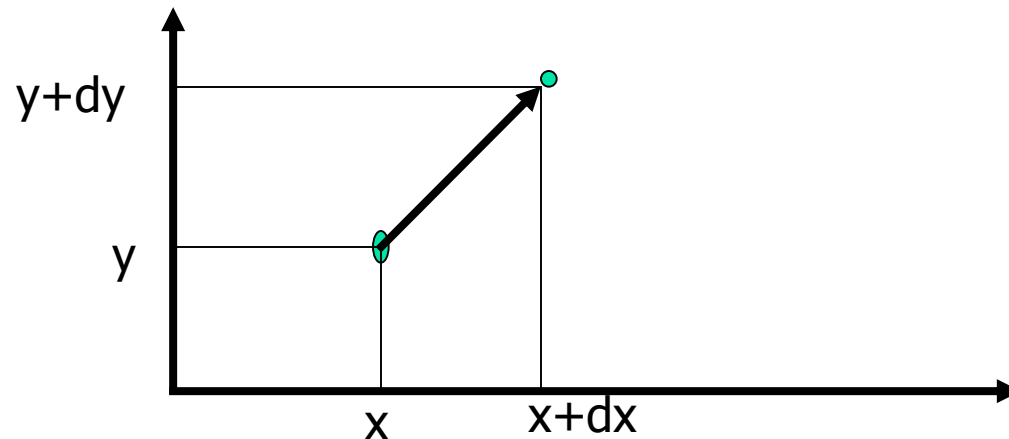
- Đối tượng là sự đóng gói của dữ liệu và phương thức.
- Đối tượng = Dữ liệu + Phương thức
(object = data + method)
 - Dữ liệu: Mô tả đối tượng.
 - Phương thức: Các hàm xử lý của đối tượng.
- Mỗi đối tượng có dữ liệu riêng và phương thức riêng.

Chương 2. Lớp và đối tượng

Đối tượng



- Ví dụ 3.1. Một mô tả về 1 đối tượng điểm như sau:
 - Một đối tượng được xác định bởi cặp tọa độ (x,y) .



Chương 2. Lớp và đối tượng

Đối tượng



- Các thao tác tác động lên đối tượng điểm gồm:
 - Hàm khởi tạo điểm ở toạ độ (ox, oy) ;
 - Hàm tịnh tiến điểm có toạ độ (x,y) đến điểm có toạ độ $(x+dx, y+dy)$.
 - Hàm hiển thị toạ độ điểm.
- Đối tượng điểm có thể được mô tả như sau:
 - Dữ liệu:
 - Cặp toạ độ (x,y) .
 - Phương thức:
 - Hàm khởi tạo toạ độ điểm.
 - Hàm tịnh tiến.
 - Hàm hiển thị toạ độ của điểm.

Chương 2. Lớp và đối tượng

Đối tượng

- Khai báo về dữ liệu:
 - `float x,y;`
- Phương thức xử lý dữ liệu:
 - Hàm đặt tọa độ của điểm tại (ox,oy)

```
void init(float ox, float oy) {  
    x = ox;  
    y = oy;  
}
```

Chương 2. Lớp và đối tượng

Đối tượng



- Hàm tịnh tiến tọa độ điểm:

```
void move(float dx, float dy) {  
    x += dx;  
    y += dy;  
}
```
- Hàm hiển thị tọa độ điểm:

```
void display() {  
    System.out.println("x="+x);  
    System.out.println("y="+y);  
};
```

Chương 2. Lớp và đối tượng

Đối tượng



■ Bài tập 2.1:

- Hãy mô tả bài toán tính diện tích và chu vi đường tròn về dạng mô tả 1 đối tượng đường tròn.
- Hãy mô tả bài toán tính diện tích và chu vi hình chữ nhật về dạng mô tả 1 đối tượng hình chữ nhật.
- Hãy mô tả bài toán giải phương trình bậc nhất về dạng mô tả 1 đối tượng phương trình bậc nhất.

Chương 2. Lớp và đối tượng

Lớp (class)



- Một mô tả chung cho các đối tượng cùng loại gọi là lớp.
- Lớp là mô tả tổng quát của đối tượng, đối tượng là 1 thể hiện cụ thể (instance) của lớp.
- Trong Java, lớp được định nghĩa như 1 cấu trúc nhưng có thêm các hàm thành phần.

Chương 2. Lớp và đối tượng

Lớp



■ Khai báo lớp:

```
<quyền truy cập> class <className>{  
    <kiểu dữ liệu> field1;  
    <kiểu dữ liệu> field2;  
    Constructor  
    Method_1  
    Method_2  
};
```



- Trong đó:
 - class: từ khóa của Java
 - className: được đặt theo quy tắc đặt tên biến, nên là 1 danh từ.
 - Đối với field: Khai báo như khai báo biến.
 - mothod1, method2: là các phương thức/hàm thể hiện các thao tác xử lý, tác động lên các thành phần dữ liệu lớp.
 - Thứ tự khai báo dữ liệu và các method là không quan trọng.
 - constructor: là sự xây dựng, khởi tạo đối tượng lớp

Khai báo lớp



■ Ví dụ 3.2. Lớp các điểm trên mặt phẳng

```
package BaithuchanhSo2;  
public class Point {  
    private float x,y;  
    public void init(float ox, float oy){  
        x=ox;  
        y=oy;  
    }  
    public void move(float dx, float dy){  
        x +=dx;  
        y +=dy;  
    }  
}
```

Chương 2. Lớp và đối tượng

Khai báo lớp



```
public void display(){
    System.out.println("x= "+x+" y= "+y);
}
public static void main(String[] args) {
    Point p=new Point();
    p.init(2, 3);
    p.display();
    p.move(3, 4);
    p.display();
}
}
```

Chương 2. Lớp và đối tượng

Lớp



■ Tạo đối tượng:

- Dùng từ khóa new:

<Tên lớp> <Tên đối tượng> = new <Tên lớp>();

- Mỗi đối tượng có 1 tập các thành phần riêng.
- Ví dụ: point p=new point();

p
- x
- y
- init
- move
- display

Chương 2. Lớp và đối tượng

Lớp



Ví dụ: Xây dựng các lớp mô tả các đối tượng:

- Đối tượng sinh viên bao gồm:
 - tập thuộc tính (mã sinh viên, họ, tên, ...)
 - Tập phương thức (nhập mới thông tin sv, sửa, xoá, xuất(xem) thông tin sv...)
- Đối tượng thí sinh bao gồm
 - Thuộc tính: sbd, họ, tên, điểm môn 1, điểm môn 2, điểm môn 3
 - Phương thức: nhập, xuất, tính tổng điểm, xem kết quả,...

Chương 2. Lớp và đối tượng

Xác định phạm vi truy xuất



- Phạm vi truy xuất là khả năng truy nhập của một hàm, một lớp nào đó đến các thành phần của lớp.
- Mọi thành phần liệt kê trong phần **public** của lớp truy xuất được ở bất kỳ các hàm, các lớp khác.
- Mọi thành phần liệt kê trong phần **private** của lớp chỉ truy xuất được trong lớp.

Phạm vi truy xuất



- **Private:** Thành viên trong lớp A được đánh dấu là **private** thì chỉ được truy cập bởi các phương thức của lớp A
- **Public** (mặc định-Java): không hạn chế. Những thành phần được đánh dấu **public** có thể được dung bởi bất kì các phương thức của lớp bao gồm những lớp khác.
- **Protected:** Thành viên trong lớp A được đánh dấu là **protected** thì chỉ được các phương thức bên trong lớp A và những phương thức dẫn xuất từ lớp A truy cập.

Chương 2. Lớp và đối tượng

Khả năng của hàm thành phần



- Cho phép định nghĩa chồng hàm thành phần.
- Cho phép sử dụng tham số ngầm định.
- Cho phép sử dụng đối tượng làm tham số hàm thành phần.
- Cho phép đối tượng làm giá trị trả về.

Chương 2. Lớp và đối tượng

Hàm thiết lập (constructor)



- Khi 1 đối tượng được tạo ra, nó luôn gọi đến 1 hàm thiết lập của nó.
- Chức năng hàm thiết lập:
 - Khởi tạo các thành phần dữ liệu.
 - Xin cấp phát bộ nhớ cho các thành phần dữ liệu động.
- Các quy định xây dựng hàm thiết lập:
 - Hàm thiết lập có tên trùng với tên lớp.
 - Hàm thiết lập phải có quyền public.

Chương 2. Lớp và đối tượng

Hàm thiết lập (constructor)



- Cú pháp hàm thiết lập:

```
access_modifier <ClassName> ([list of parameters]){  
    // body of the method  
}
```

Chương 2. Lớp và đối tượng

Hàm thiết lập

- Hàm không có giá trị trả về và không cần từ khoá void.
- Một lớp có thể định nghĩa nhiều hàm thiết lập. Khi định nghĩa nhiều hàm thiết lập có thể khai báo đối tượng theo nhiều dạng khác nhau.
- Hàm thiết lập có thể khai báo với các tham số có giá trị ngầm định.
- Khi không có hàm thiết lập được định nghĩa, lớp sử dụng hàm thiết lập ngầm định (hàm không có tham số và không làm gì cả).

Hàm thiết lập

Ví dụ 3.3. Xây dựng lớp point sử dụng hàm thiết lập.

- Hàm thiết lập không tham số

```
public Point() {  
    }  
}
```

- Hàm thiết lập 2 tham số

```
public Point(float x, float y) {  
    this.x = x;  
    this.y = y;  
}
```

Chương 2. Lớp và đối tượng

Hàm thiết lập

```
package BaithuchanhSo2;
public class Point {
    private float x,y;

    public Point(float x, float y) {
        this.x = x;
        this.y = y;
    }
    public Point() {
    }
    public void move(float dx, float dy){
        x +=dx;
        y +=dy;
    }
}
```

Chương 2. Lớp và đối tượng

Hàm thiết lập

```
public void display(){
    System.out.println("x= "+x+" y= "+y);
}
public static void main(String[] args) {
    Point p=new Point(2,3);
    p.display();
    p.move(3, 4);
    p.display();
}
```


Chương 2. Lớp và đối tượng

Hàm thiết lập



- Ví dụ 3.5. Cho 2 khai báo lớp:

```
class A{  
    ...  
    public A() {};  
    public A(int x){...};  
};
```

```
class B{  
    ...  
    //Không có hàm thiết lập.  
};
```

- Lớp A sử dụng hàm thiết lập nào ?, lớp B sử dụng hàm thiết lập nào?

Chương 2. Lớp và đối tượng

Hàm thiết lập



- Ví dụ 2.6. Cho biết kết quả khi chạy chương trình sau:

```
class Customer{ String name;
public Customer(String name) {
this.name = name; } }
public class Entry {
public static void main(String[] args) {
    Customer customer = new Customer();
    System.out.println("Welcome to
Codelearn!"); }
}
```

Chương 2. Lớp và đối tượng

Hàm thiết lập



- **Ví dụ 2.6** do đã tạo hàm thiết lập có tham số, nên hàm thiết lập ngầm định khôn được tạo ra. Trong main khởi tạo đối tượng không tham số nên báo lỗi. Cách sửa:

```
class Customer{
    String name;
    public Customer() {
    }
    public Customer(String name) {
        this.name = name; }
}
public class Entry {
    public static void main(String[] args) {
        Customer customer = new Customer();
        System.out.println("Welcome to Codelearn!"); }
}
```

Chương 2. Lớp và đối tượng

Toán tử this



- Hoạt động giống 1 con trỏ để trỏ vào đối tượng gần nhất chứa nó.
- Nó có thể gọi đến các thuộc tính và phương thức của một đối tượng.
- Trong 1 phương thức, tham số được ưu tiên nhắc tới hơn là thuộc tính.

Chương 2. Lớp và đối tượng

Toán tử this



```
public Point(float x, float y) {  
    x = x;  
    y = y;  
}
```

//hai x và hai y đều được hiểu là
//tham số của phương thức

```
public Point(float x, float y) {  
    this.x = x;  
    this.y = y;  
}
```

```
public static void main(String[] args) {  
    Point p=new Point(2,3);  
    p.display();  
    p.move(3, 4);  
    p.display();  
}
```

x= 0.0 y= 0.0
x= 3.0 y= 4.0

x= 2.0 y= 3.0
x= 5.0 y= 7.0

Chương 2. Lớp và đối tượng

Toán tử this



```
class A{  
    int songuyen;  
    class B{  
        int soNguyen;  
        public void tinhTong(){  
            int tong=this.soNguyen+A.this.soNguyen;  
        }  
    }  
}
```

//this.soNguyen: truy cập tới thuộc tính soNguyen của đối tượng B

//A.this.soNguyen: truy cập tới thuộc tính soNguyen của đối tượng A

Chương 2. Lớp và đối tượng

Getter và Setter



```
package BaithuchanhSo2;
import java.util.Scanner;
public class HinhChuNhat {
    int dai, rong;
    public HinhChuNhat(int dai, int rong) {
        this.dai = dai;
        this.rong = rong;
    }
    public HinhChuNhat() {
    }
```

Chương 2. Lớp và đối tượng



```
public int getDai() {  
    return dai;  
}  
public int getRong() {  
    return rong;  
}
```


Chương 2. Lớp và đối tượng



```
public void setDai(int dai) {  
    do{  
        System.out.println("Moi nhap gia tri cho chieu dai :");  
        Scanner sr=new Scanner(System.in);  
        dai=sr.nextInt();  
        if(dai<0) System.out.println("Moi nhap lai gia tri >0");  
    }  
    while (dai<0);  
    this.dai = dai;  
}
```

Chương 2. Lớp và đối tượng



```
public void setRong(int rong) {  
    do {  
        System.out.println("Moi nhap gia tri cho chieu rong:");  
        Scanner sr=new Scanner(System.in);  
        rong=sr.nextInt();  
        if(rong<0) System.out.println("Moi nhap lai gia trị >0");  
    } while (rong <0);  
    this.rong = rong;  
}
```

Chương 2. Lớp và đối tượng



```
public void chuVi(){  
    System.out.println("Chu vi của hcn la" + (dai + rong) * 2);  
}  
public void dienTich(){  
    System.out.println("Dien tich của hcn la" + dai * rong);  
}
```

Chương 2. Lớp và đối tượng



```
public static void main(String[] args) {  
    HìnhChuNhat a=new HìnhChuNhat();  
    a.setDai(0);  
    a.setRong(0);  
    a.chuVi();  
    a.dienTich();  
}  
}
```

Chương 2. Lớp và đối tượng

Hàm huỷ bỏ (Destructor)



Trong Java, các đối tượng cấp phát động bằng toán tử new, khi không tồn tại biến nào trỏ đến đối tượng, đối tượng đó xem như không còn cần đến nữa và bộ nhớ dành cho nó có thể được tự động giải phóng bởi bộ thu gom rác (Garbage Collector).

Trình thu gom rác hoạt động trong một tuyến đoạn (Thread) độc lập với chương trình của bạn. Bạn không phải bận tâm gì với công việc này.

Chương 2. Lớp và đối tượng

Hàm huỷ bỏ



- Java cũng cho phép ta viết hàm hủy
- Hàm hủy trong Java chỉ được gọi bởi trình thu gom rác, bạn khó đoán trước vào lúc nào hàm hủy sẽ được gọi, do đó hàm hủy ít được sử dụng.
- Dạng hàm hủy như sau :

```
protected void finalize() {  
    // Body of Method  
}
```

Chương 2. Lớp và đối tượng

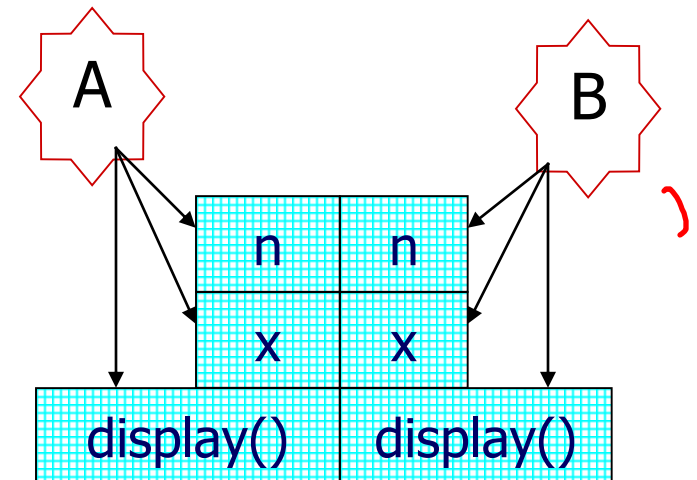
Biến tĩnh

DL
T i

- Trong 1 chương trình, các đối tượng thuộc cùng 1 lớp sở hữu các thành phần dữ liệu riêng và các hàm riêng của nó.
- Ví dụ khai báo lớp:

```
class abc{  
    private  
        int n;  
        float x;  
    public  
        void display();  
};
```

Khai báo: abc A, B;



Chương 2. Lớp và đối tượng

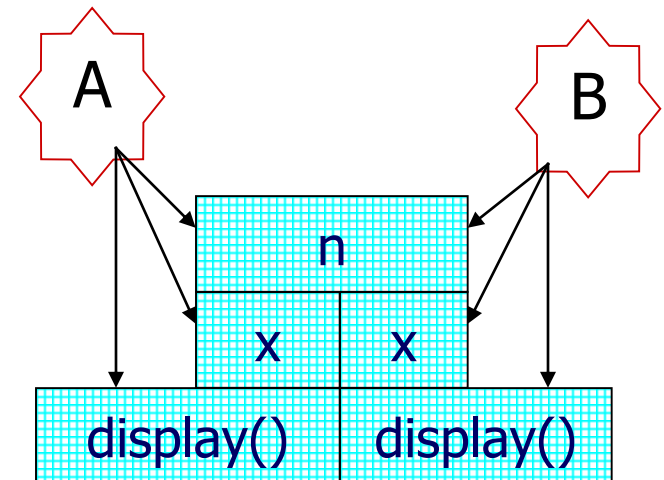
Biến tĩnh



- Để chia sẻ các thành phần chung của các đối tượng, đặt khoá **static** trước thành phần khai báo.
- Ví dụ với khai báo lớp:

```
class abc{  
    private  
        static int n;  
        float x;  
    public  
        void display();  
};
```

Khai báo: `abc A, B;`



Chương 2. Lớp và đối tượng

Biến tĩnh



```
class Student{
    int rollno;
    String name;
    String college="SGU";
}
/* bao nhiêu đối tượng
được tạo ra sẽ tạo ra bấy
nhiều ô nhớ lưu tất cả các
thuộc tính, mặc dù có
những thuộc tính chung
cho tất cả accs đối tượng
*/
```

```
public class Student1 {
    int rollno;
    String name;
    static String college = "SGU";

    Student8(int r, String n) {
        rollno = r;
        name = n;    }
    void display() {
        System.out.println(rollno + " - " + name + " - " + college);
    }
    public static void main(String args[]) {
        Student1 s1 = new Student8(111, "Thông");
        Student1 s2 = new Student8(222, "Minh");
        s1.display();
        s2.display();    } }
```

/*dù cho bao nhiêu đối tượng được tạo ra thì chỉ có 1 ô nhớ
dung để lưu trữ biến college. Tiết kiệm ô nhớ */

Chương 2. Lớp và đối tượng

Biến tĩnh



```
public class Counter1 {  
    int count = 0; /*sẽ lấy bộ nhớ khi  
instance được tạo ra*/
```

```
    Counter1() {  
        count++;  
        System.out.println(count);  
    }
```

```
    public static void main(String  
args[]) {  
  
        Counter1 c1 = new Counter1();  
        Counter1 c2 = new Counter1();  
        Counter1 c3 = new Counter1();  
    } }
```

Kết quả::

1

1

1

```
public class Counter2 {  
    static int count = 0; /* sẽ lấy bộ nhớ chỉ  
một lần*/
```

```
    Counter2() {  
        count++;  
        System.out.println(count);  
    }
```

```
    public static void main(String args[]) {  
  
        Counter2 c1 = new Counter2();  
        Counter2 c2 = new Counter2();  
        Counter2 c3 = new Counter2();  
    } }
```

1

2

3

Chương 2. Lớp và đối tượng

Phương thức tĩnh



- Một phương thức static thuộc lớp chứ không phải đối tượng của lớp.
- Một phương thức static gọi mà không cần tạo một instance của một lớp.

Quy tắc gọi phương thức thường

Tên_đối_tượng.phương_thức([tham số])

Quy tắc gọi phương thức static

Tên_lớp.phương_thức_static([tham số])

- Phương thức static có thể truy cập biến static và có thể thay đổi giá trị của nó.

Chương 2. Lớp và đối tượng

Phương thức tĩnh



```
public class Student {  
    int rollno;  
    String name;  
    static String college = "SP Kỹ thuật";  
    static void change() {  
        college = "SGU"; // Thay đổi thuộc tính static (thuộc tính chung của tất cả các đối tượng)    }  
    Student(int r, String n) {  
        rollno = r;  
        name = n;    }  
    void display() {  
        System.out.println(rollno + " - " + name + " - " + college);    }  
    public static void main(String args[]) {  
        Student.change();  
        Student s1 = new Student(111, "Thông");  
        Student s2 = new Student(222, "Minh");  
        Student s3 = new Student(333, "Anh");  
        s1.display();  
        s2.display();  
        s3.display();    }  
}
```

Kết quả ct?

Chương 2. Lớp và đối tượng

Phương thức tĩnh




- Có hai hạn chế chính đối với phương thức static:
 - Phương thức static không thể sử dụng biến non-static hoặc gọi trực tiếp phương thức non-static.
 - Từ khóa this và super không thể được sử dụng trong ngữ cảnh static.
- Ví dụ:

```
class A {  
    int a = 40; // non static  
    public static void main(String args[]) {  
        System.out.println(a);  
    }  
}
```

Chương 2. Lớp và đối tượng

Phương thức tĩnh



Ans: Tại sao phương thức main của Java phải luôn là static?

Bởi vì không cần thiết phải tạo đối tượng để gọi phương thức static. Nếu nó là phương thức non-static, JVM đầu tiên tạo đối tượng và sau đó gọi phương thức main() mà có thể gây ra vấn đề về cấp phát bộ nhớ phụ.

Chương 2. Lớp và đối tượng

Thành phần tĩnh



- Ví dụ: Tạo 1 lớp student mô tả các đối tượng sinh viên, lớp gồm:
 - Thuộc tính name mô tả tên sinh viên.
 - Thuộc tính fee mô tả tiền học phí.
 - Thuộc tính total mô tả tổng học phí đã nhập.
 - Viết chương trình nhập thông tin và tạo 3 đối tượng sinh viên, tính tổng học phí của 3 sinh viên đã tạo, hiển thị các thông tin về sinh viên.

Chương 2. Lớp và đối tượng

Thành phần tĩnh



```
class student{
    private:
        char *name;
        float fee;
        static float total;
    public:
        student(char *n = NULL, float f=0){
            name = strdup(n);
            fee=f;
            total += fee;
        }
        void Display() {
            cout<<"\n Name:"<<name<<"\t Fee:"<<fee;
            cout<<"\n Total:"<<total;
        }
};
```


Chương 2. Lớp và đối tượng

Thành phần tĩnh

//Khởi tạo thành phần dữ liệu tĩnh ở ngoài lớp, ngoài các hàm.

float student::total=0;

//Chương trình chính

```
void main() {  
    student s1("a",10), s3("c",30);  
    s1.Display();  
    student s2("b",20);  
    s2.Display();  
    s3.Display();  
    getch();  
}
```

Chương 2. Lớp và đối tượng

Tóm tắt



- Đối tượng
- Khai báo lớp, các thành phần dữ liệu, hàm.
- Hàm thiết lập và hàm huỷ bỏ
- Các thành phần tĩnh