Bài 4

Các kỹ thuật xây dựng lớp

Nội dung

1. Đóng gói
2. Hàm khởi tạo
3. Nạp chồng
4. Kết tập

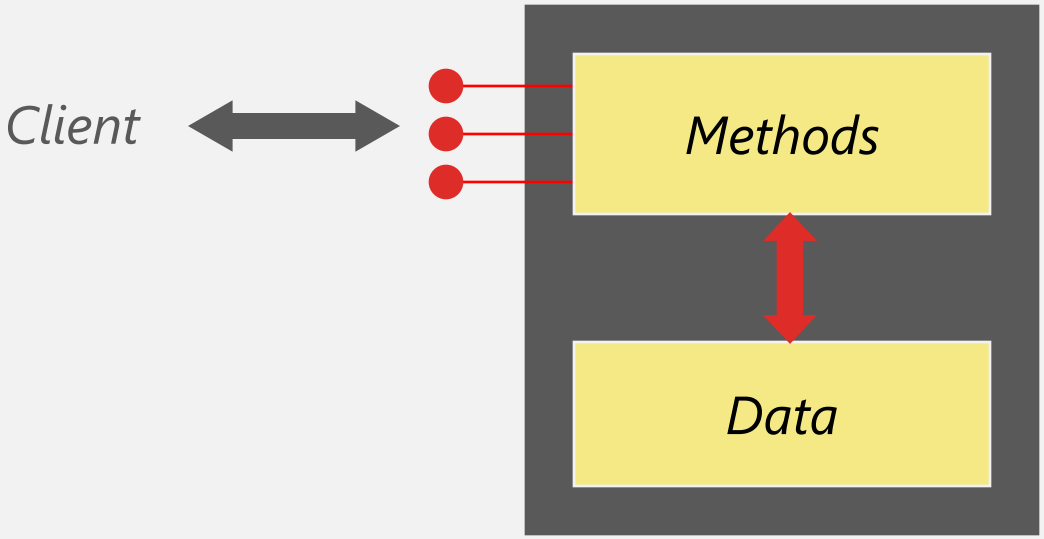
1

Đóng gói

Encapsulation

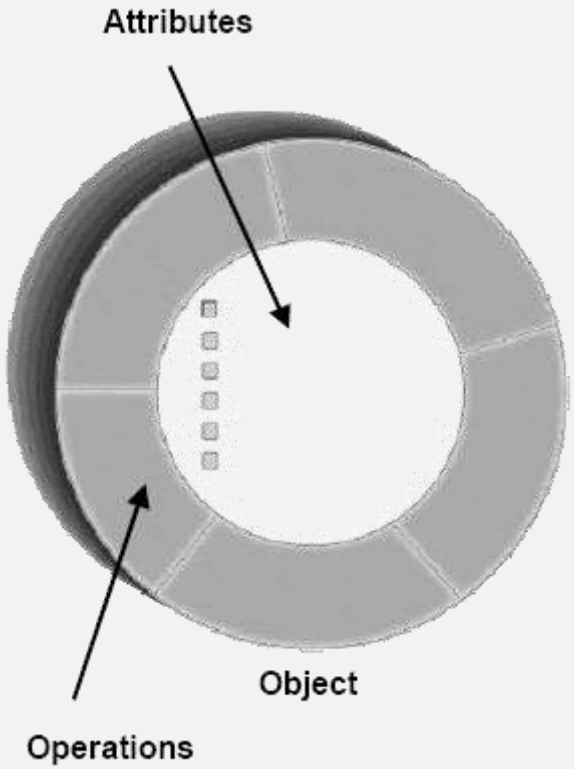
Đóng gói

- Một đối tượng có hai khung nhìn:
  - Bên trong: Chi tiết về các thuộc tính và các phương thức của lớp tương ứng với đối tượng
  - Bên ngoài: Các dịch vụ mà một đối tượng có thể cung cấp và cách đối tượng đó tương tác với phần còn lại của hệ thống



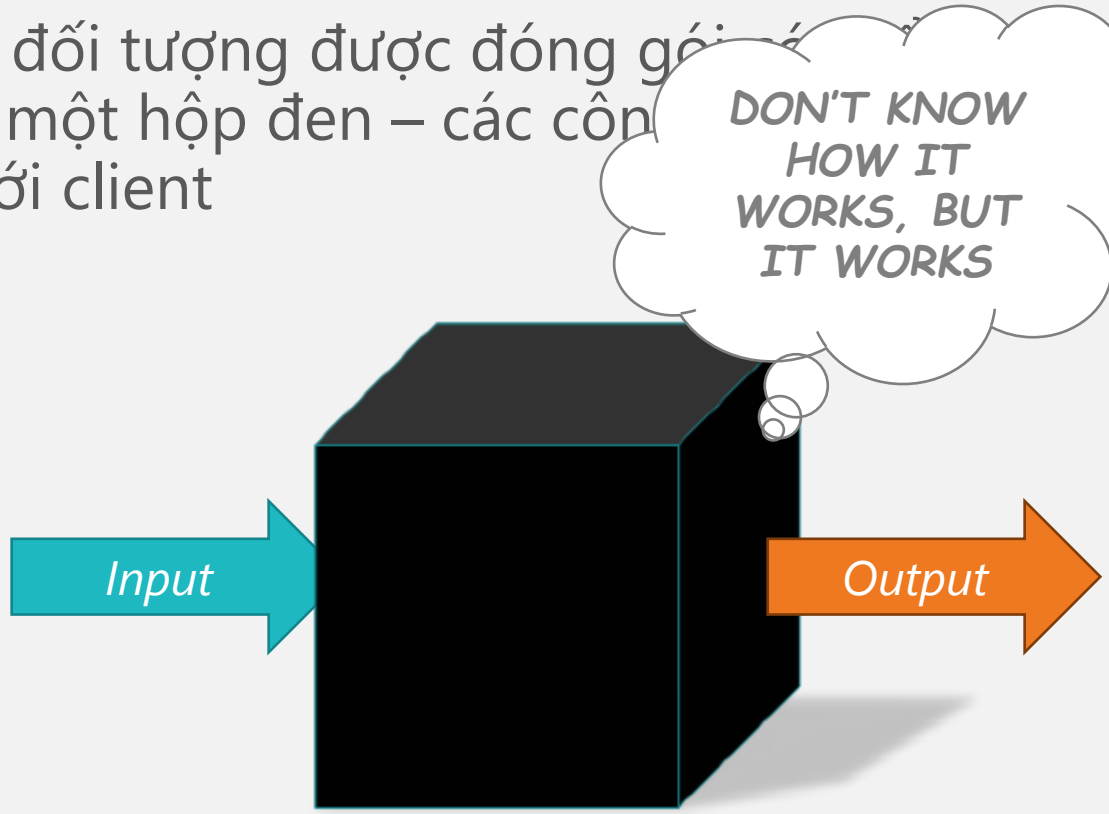
Đóng gói

- Đóng gói (Encapsulation)
  - Dữ liệu và phương thức được đóng gói trong một lớp
  - Dữ liệu được che giấu ở bên trong lớp và chỉ được truy cập và thay đổi ở các phương thức bên ngoài



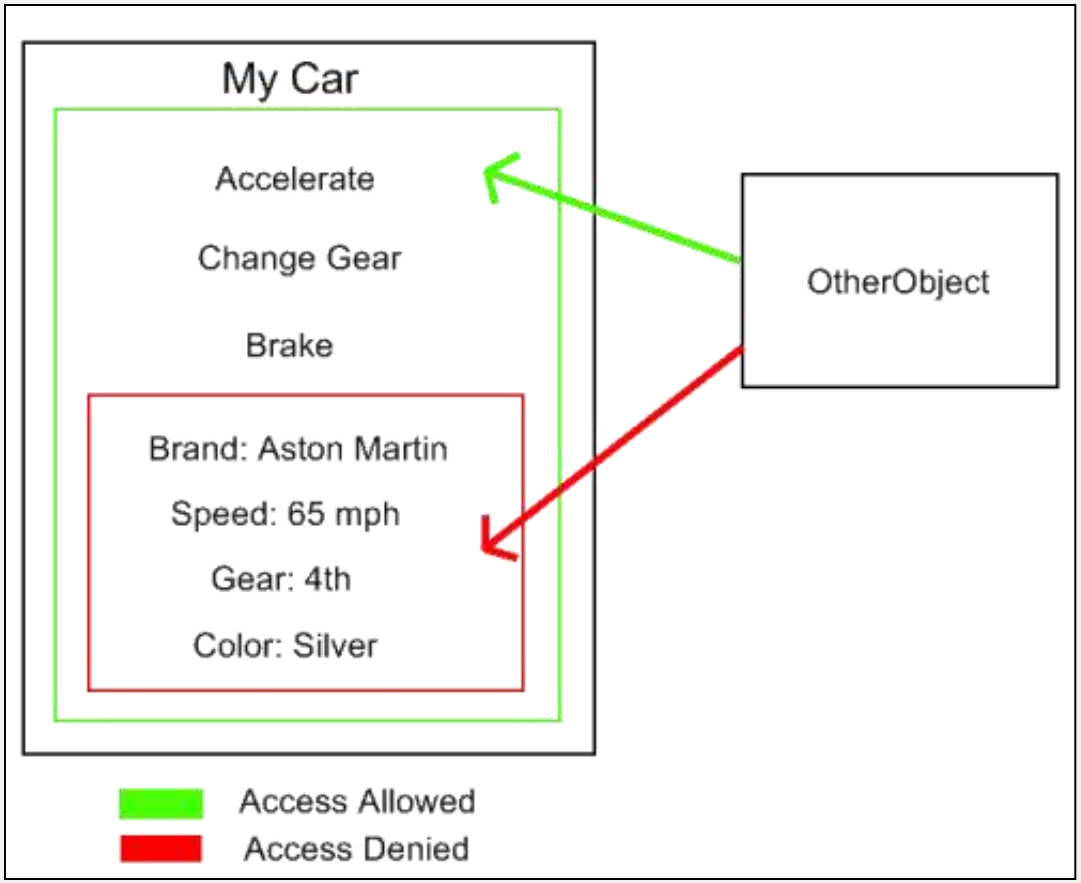
Đóng gói

- Một đối tượng là một thực thể được đóng gói, cung cấp tập các dịch vụ nhất định
- Một đối tượng được đóng gói [illegible] như một hộp đen – các chi tiết [illegible] ẩn so với client



Che giấu dữ liệu

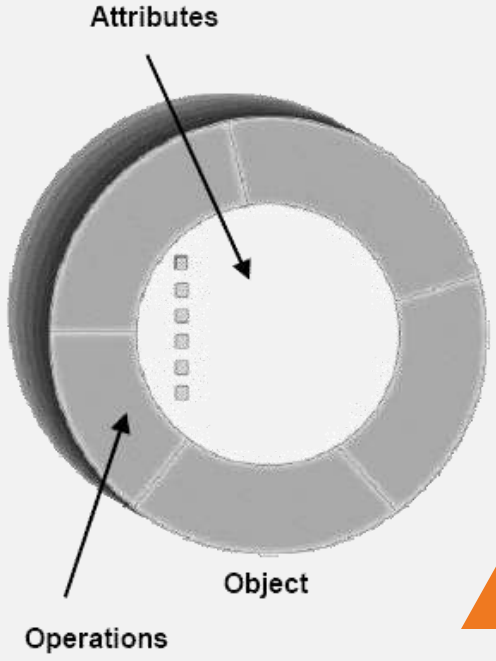
- Dữ liệu được che giấu ở bên trong lớp và chỉ được truy cập và thay đổi ở các phương thức bên ngoài
  - Tránh thay đổi trái phép hoặc làm sai lệch dữ liệu



Cơ chế che giấu dữ liệu

- Các thành viên dữ liệu
  - Chỉ có thể truy cập từ các phương thức bên trong lớp
  - Chỉ định truy cập là private để bảo vệ dữ liệu
- Các đối tượng khác muốn truy nhập vào dữ liệu riêng tư này phải thông qua các phương thức public

BankAccount
- owner: String - balance: double
+ debit(double): boolean +credit(double)



Cơ chế che giấu dữ liệu

- Các thành phần dữ liệu là private → Để truy cập và chỉnh sửa các giá trị của dữ liệu, lớp cần phải cung cấp các dịch vụ
  - Accessor (getter): Trả về giá trị hiện tại của một thuộc tính (dữ liệu)
  - Mutator (setter): Thay đổi giá trị của một thuộc tính
  - Thường là getX và setX, trong đó x là tên thuộc tính

```
package com.megabank.models;

public class BankAccount {
  private String owner;
  private double balance = 0.0;
}
```

```
public String getOwner() {
  return owner;
}
```

Phương thức Get

- Các phương thức truy vấn (query method, accessor) là các phương thức dùng để hỏi về giá trị của các thành viên dữ liệu của một đối tượng
- Có nhiều loại câu hỏi truy vấn có thể:
  - truy vấn đơn giản (“giá trị của x là bao nhiêu?”)
  - truy vấn điều kiện (“thành viên x có lớn hơn 10 không?”)
  - truy vấn dẫn xuất (“tổng giá trị của các thành viên x và y là bao nhiêu?”)
- Đặc điểm quan trọng của phương thức truy vấn là nó không nên thay đổi trạng thái hiện tại của đối tượng
  - không thay đổi giá trị của thành viên dữ liệu nào.

Phương thức Set

- Các phương thức thiết lập (Mutator, setter) là các phương thức dùng để thay đổi giá trị các thành viên dữ liệu
- Ưu điểm của việc sử dụng các phương thức setter là có thể sử dụng các phương thức setter để đảm bảo tính hợp lệ của các thành phần dữ liệu
  - Kiểm tra giá trị đầu vào trước khi gán vào các thuộc tính

Ví dụ

restricted access: *private* members are *not externally accessible*; but we need to know and modify their values

set methods: *public* methods that allow clients to *modify private* data; also known as *mutators*

```
public class Time {
    private int hour;
    private int minute;
    private int second;

    public Time () {
        setTime(0, 0, 0);
    }

    public void setHour (int h) { hour = ( ( h >= 0 && h < 24 ) ? h : 0 ); }

    public void setMinute (int m) { minute = ( ( m >= 0 && m < 60 ) ? m : 0 ); }

    public void setSecond (int s) { second = ( ( s >= 0 && s < 60 ) ? s : 0 ); }

    public void setTime (int h, int m, int s) {
        setHour(h);
        setMinute(m);
        setSecond(s);
    }

    public int getHour () { return hour; }

    public int getMinute () { return minute; }

    public int getSecond () { return second; }
}
```

get methods: *public* methods that allow clients to *read private* data; also known as *accessors*

2

Hàm khởi tạo

Constructor

Khởi tạo đối tượng

- Để khởi tạo đối tượng: sử dụng từ khóa **new**

```
BankAccount acc = new BankAccount();
```

phương thức khởi tạo (ngầm định)

- Phương thức khởi tạo ngầm định (implicit) sẽ tự động gán các giá trị mặc định cho các thành phần dữ liệu
  - Được thực hiện trước khi lập trình viên có thể tác động lên đối tượng
  - Nếu không muốn sử dụng phương thức khởi tạo ngầm định → có thể viết phương thức khởi tạo cụ thể (explicit) cho lớp

Phương thức khởi tạo

- Là phương thức được gọi để gán các giá trị cho các thành phần dữ liệu khi đối tượng được khởi tạo
  - Tên của phương thức trùng tên lớp
  - KHÔNG có kiểu dữ liệu trả về
  - Có thể có hoặc không có tham số
- Phương thức khởi tạo KHÔNG được coi là thành viên của lớp
- Java sẽ không sử dụng phương thức khởi tạo ngầm định khi lập trình viên đã viết phương thức khởi tạo cho lớp

Phương thức khởi tạo

- Ví dụ phương thức khởi tạo không có tham số (còn gọi là phương thức khởi tạo mặc định)

```
class BankAccount {
  private String owner;
  private long balance;
  public BankAccount() {
    this.name = "NONAME"; this.balance = 0;
  }
}
```

Phương thức khởi tạo

- Ví dụ phương thức khởi tạo có tham số

```
class BankAccount {
  private String owner;
  private long balance;
  public BankAccount(String name, double balance) {
    this.name = name; this.balance = balance;
  }
}
```

- Mục đích: giúp khởi tạo đối tượng dễ dàng hơn

VD:
```
BankAccount account =
        new BankAccount("Nguyen Van A", 100000);
```

Khởi tạo đối tượng trong C++

- Phương thức khởi tạo: Tương tự Java
- Khởi tạo đối tượng
  - Đối với đối tượng

```
BankAccount acc;
BankAccount acc("Tran Van A", 100000);
```

  - Đối với con trỏ đối tượng

```
BankAccount *acc = new BankAccount();
BankAccount *acc = new BankAccount("Tran Van A", 100000);
```

Hàm hủy C++

- Ngược lại với quá trình khởi tạo đối tượng, khi giải phóng đối tượng chúng ta phải giải phóng toàn bộ bộ nhớ đã được cấp phát cho đối tượng. Chức năng của hàm huỷ (destructor) sẽ thực hiện vai trò này
- Ví dụ:

```
class  A {
int n;
public:
  A();  // constructor
 ~A();  // destructor
};
```

- Java: không dùng hàm hủy.

Ví dụ 1

```
public class BankAccount {
  private String owner;
  private double balance;
}
public class Test {
  public static void main(String args[]) {
    BankAccount acc1 = new BankAccount();
  }
}
```

→ Phương thức khởi tạo mặc định do Java cung cấp.

Ví dụ 2

```
public class BankAccount {
  private String owner;
  private double balance;
  public BankAccount() {
    owner = ”noname”;
  }
}
public class Test {
  public static void main(String args[]) {
    BankAccount acc1 = new BankAccount();
  }
}
```

→ Phương thức khởi tạo mặc định tự viết.

Ví dụ 3

```
public class BankAccount {
  private String owner;
  private double balance;
  public BankAccount(String name) {
    owner = name;
  }
}
public class Test {
  public static void main(String args[]) {
     BankAccount account1 = new
       BankAccount(); //Error
     BankAccount account2 = new
       BankAccount("Hoang");
  }
}
```

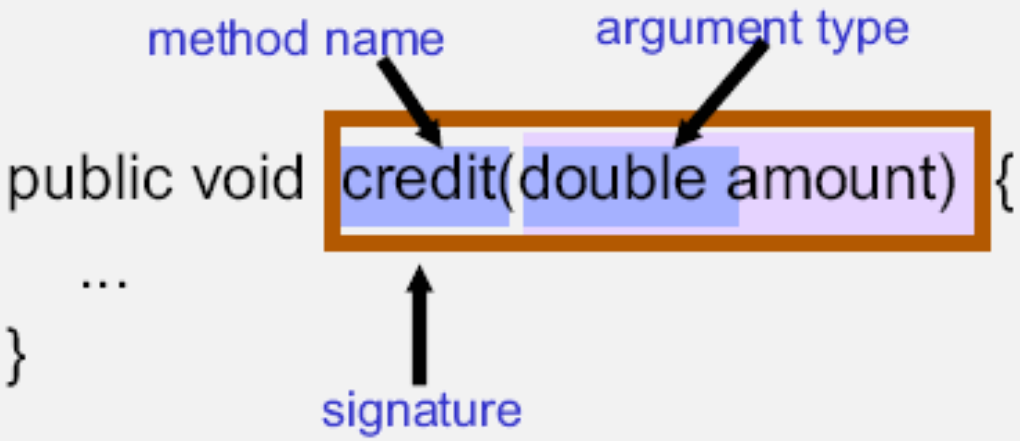
3

Nạp chồng

Overload

Chữ ký phương thức

- Chữ ký của phương thức bao gồm:
  - Tên phương thức
  - Số lượng các đối số và kiểu của chúng



Nạp chồng

- Nạp chồng hay chồng phương thức (method overloading): Các phương thức trong cùng một lớp có thể trùng tên nhưng khác chữ ký:
  - Số lượng tham số khác nhau
  - Nếu cùng số lượng tham số thì kiểu dữ liệu các tham số phải khác nhau
- Mục đích:
  - Tên trùng nhau để mô tả bản chất công việc
  - Thuận tiện cho lập trình vì không cần phải nhớ quá nhiều tên phương thức mà chỉ cần nhớ một tên và lựa chọn các tham số cho phù hợp.

Ví dụ

```
class MyDate {
   int year, month, day;
   public boolean setMonth(int m) { …}
   public boolean setMonth(String s) { …}
}

public class Test{
  public static void main(String args[]){
   MyDate d = new MyDate();
   d.setMonth(9);
   d.setMonth(”September”);
   }
}
```

Ví dụ về nạp chồng trong Java

- Trong Java:
  - Phương thức `println()` trong `System.out.println()` có 10 khai báo với các tham số khác nhau: `boolean`, `char[]`, `char`, `double`, `float`, `int`, `long`, `Object`, `String`, và một không có tham số.
  - Không cần sử dụng các tên khác nhau (chẳng hạn `printString` hoặc `printDouble`) cho mỗi kiểu dữ liệu muốn hiển thị.

Chú ý

- Các phương thức chỉ được xem xét là chồng khi chúng thuộc cùng một lớp
- Chỉ nên sử dụng kỹ thuật này với các phương thức có cùng mục đích, chức năng; tránh lạm dụng
- Khi dịch, trình dịch căn cứ vào số lượng hoặc kiểu dữ liệu của tham số để quyết định gọi phương thức nào phù hợp.

  → Nếu không chọn được hoặc chọn được nhiều hơn 1 phương thức thì sẽ báo lỗi.

Ví dụ

```
void prt(String s) {
System.out.println(s); }
void f1(char x) { prt("f1(char)"); }
void f1(byte x) { prt("f1(byte)"); }
void f1(short x) { prt("f1(short)"); }
void f1(int x) { prt("f1(int)"); }
void f1(long x) { prt("f1(long)"); }
void f1(float x) { prt("f1(float)"); }
void f1(double x) { prt("f1(double)"); }
```

- Điều gì xảy ra nếu thực hiện:
  - `f1(5);`
  - `char x='a'; f1(x);`
  - `byte y=0; f1(y);`
  - `float z = 0; f1(z);`…

```
f1(int)
f1(char)
f1(byte)
f1(float)
```

Ví dụ

```
void prt(String s) {
System.out.println(s); }
void f3(short x) { prt("f3(short)"); }
void f3(int x) { prt("f3(int)"); }
void f3(long x) { prt("f3(long)"); }
void f3(float x) { prt("f3(float)"); }
```

- Điều gì xảy ra nếu thực hiện:
  - `f3(5);`
  - `char x='a'; f3(x);`
  - `byte y=0; f3(y);`
  - `float z = 0; f3(z);`
  - `f3(5.5);`

```
f3(int)
f3(int)
f3(short)
f5(float)
```

```
Error: cannot find symbol: method f3(double)
```

Chồng phương thức khởi tạo

- Trong cùng một lớp ta có thể xây dựng nhiều phương thức khởi tạo với danh sách tham số khác nhau

  →Chồng phương thức khởi tạo (constructor overloading)

- Nếu muốn gọi đến phương thức khởi tạo khác của lớp: có thể sử dụng toán tử this

```
this(danh sách tham số);
```

4

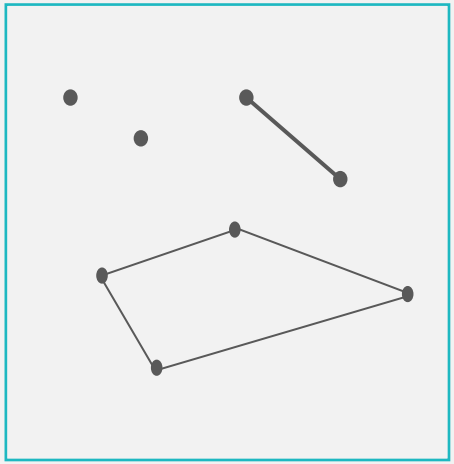
Kết tập

Aggregation

Mối quan hệ kết tập

- Lớp toàn thể chứa các đối tượng của lớp thành phần
  - Đối tượng lớp thành phần: Là một phần (is-a-part of) của lớp toàn thể
  - Quan hệ chứa/có (“has-a”) hoặc là một phần ("is-a-part-of")

- Ví dụ
  - Tứ giác gồm 4 Điểm
  - Ô tô gồm 4 Bánh xe
  - Lớp học chứa 91 Sinh viên



Ví dụ

```
class Diem {
  private int x, y;
  public Diem() {}
  public Diem(int x, int y) {
    this.x = x; this.y = y;
  }
  public void setX(int x) { this.x = x; }
  public int getX() { return x; }
  public void hienThiDiem(){
    System.out.print("("+x+", "+y+")");
  }
}
```

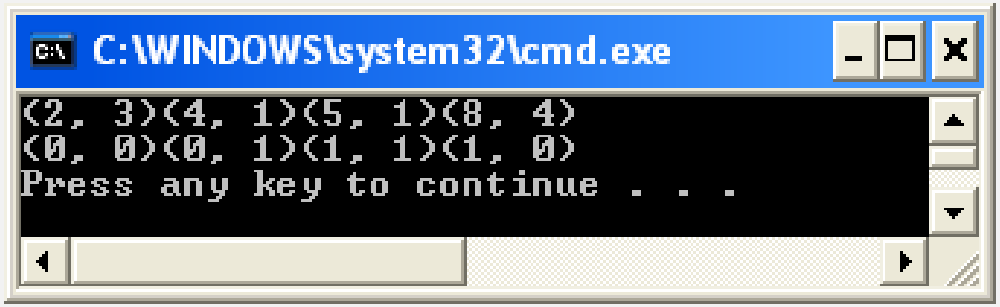
Ví dụ

```
class TuGiac {
  private Diem d1, d2;
  private Diem d3, d4;
  public TuGiac(Diem p1, Diem p2,
                Diem p3, Diem p4){
    d1 = p1; d2 = p2; d3 = p3; d4 = p4;
  }
  public TuGiac(){
    d1 = new Diem();      d2 = new Diem(0,1);
    d3 = new Diem (1,1); d4 = new Diem (1,0);
  }
  public void printTuGiac(){
    d1.printDiem();  d2.printDiem();
    d3.printDiem();  d4.printDiem();
    System.out.println();
  }
}
```

Ví dụ

```
public class Test {
  public static void main(String args[])
  {
       Diem d1 = new Diem(2,3);
       Diem d2 = new Diem(4,1);
       Diem d3 = new Diem(5,1);
       Diem d4 = new Diem(8,4);

       TuGiac tg1 = new TuGiac(d1, d2, d3, d4);
       TuGiac tg2 = new TuGiac();
        tg1.printTuGiac();
        tg2.printTuGiac();
  }
}
```



Bản chất của kết tập

- Kết tập (aggregate)
  - Các thành phần của lớp mới là các đối tượng của các lớp có sẵn.
  - Tái sử dụng các thành phần dữ liệu và các hành vi của lớp thành phần thông qua đối tượng của lớp thành phần
- Lớp mới
  - Lớp toàn thể (Aggregate/Whole)
- Lớp cũ
  - Lớp thành phần (Part).

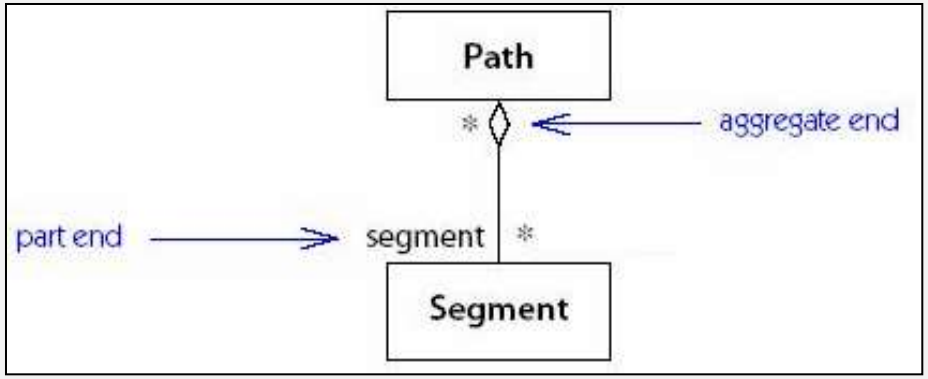
Thứ tự khởi tạo

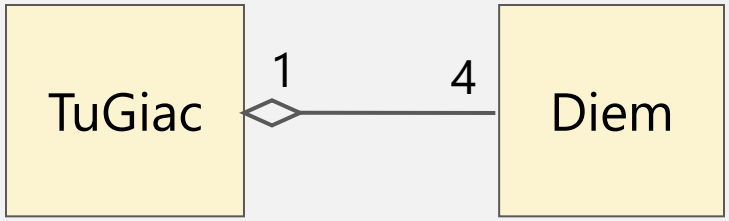
- Khi một đối tượng được tạo mới, các thuộc tính của đối tượng đó đều phải được khởi tạo và gán những giá trị tương ứng.
- Các đối tượng thành phần được khởi tạo trước

  → Các phương thức khởi tạo của các lớp của các đối tượng thành phần được thực hiện trước

Biểu diễn kết tập bằng UML

- Sử dụng "hình thoi" tại đầu của lớp toàn thể
- Sử dụng bội số quan hệ (multiplicity) tại 2 đầu
  - 1 số nguyên dương: 1, 2,...
  - Dải số (0..1, 2..4)
  - *: Bất kỳ số nào
  - Không có: Mặc định là 1





Thank you!

Any questions?