



TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM



LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI JAVA


Chương 07. Kết tập

Nguyễn Thị Thu Trang
trangntt-fit@mail.hut.edu.vn



Mục tiêu của bài học

- Sau khi kết thúc bài học, sinh viên có thể:
 - Chỉ ra được bản chất của kết tập
 - Mô tả các khái niệm cơ bản liên quan đến kết tập.
 - Biểu diễn được kết tập trên UML
 - Sử dụng các vấn đề trên với ngôn ngữ lập trình Java.




© Nguyễn Thị Thu Trang, SE-FIT-HUT

2

Chương 07. Kết tập

1. Tổng quan về kết tập
2. Biểu diễn trên UML
3. Thực thi trên Java




© Nguyễn Thị Thu Trang, SE-FIT-HUT

3

Chương 07. Kết tập

⇒ 1. Tổng quan về kết tập

2. Biểu diễn trên UML
3. Thực thi trên Java



© Nguyễn Thị Thu Trang, SE-FIT-HUT

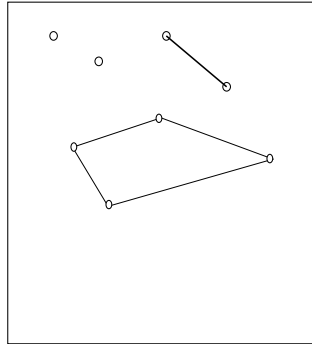
4

1. Tổng quan về kết tập

● Ví dụ:

■ Điểm

- Tứ giác gồm 4 điểm
→ Kết tập



1. Tổng quan về kết tập (2)

● Tái sử dụng mã nguồn (Re-usability)

- Tồn tại nhiều loại đối tượng có các thuộc tính và hành vi tương tự hoặc liên quan đến nhau
- Xuất hiện nhu cầu sử dụng lại các mã nguồn đã viết.
- Lớp cũ đã có, đã mất công lập trình → Sử dụng lại lớp cũ:
 - Sao chép lớp cũ thành 1 lớp khác.
 - Tạo ra lớp mới là sự kết hợp các đối tượng của lớp cũ đã có → Kết tập
 - Tạo ra lớp mới trên cơ sở phát triển từ lớp cũ đã có → Kế thừa



1. Tổng quan về kết tập (3)

● Bản chất của kết tập

- Tạo ra tham chiếu đến các đối tượng của các lớp có sẵn trong lớp mới → Lớp mới là sự kết tập các lớp cũ đã có.
 - Lớp mới chứa các tham chiếu đến các đối tượng của các lớp cũ.
 - Các tham chiếu này chính là các thành viên của lớp mới.
 - Quan hệ chứa/có ("has-a") hoặc là một phần (is-a-part-of) hoặc sử dụng ("use-a").
- Kết tập tái sử dụng thông qua lại đối tượng



Chương 07. Kết tập

1. Tổng quan về kết tập



2. Biểu diễn trên UML

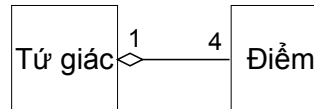
3. Thực thi trên Java



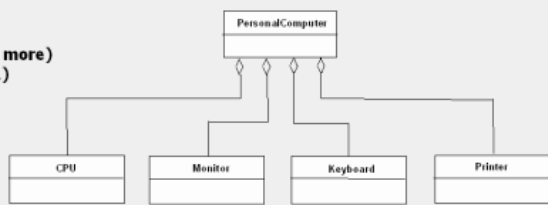
2. Biểu diễn trong UML

● Bội số quan hệ (Multiplicity)

- 1 số nguyên dương: 1, 2,...
- Dải số (0..1, 2..4)
- *: Bất kỳ số nào

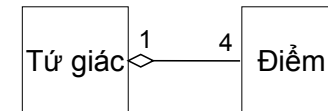


UML notation:
diamond (1 or more)
no diamond (1)



2. Biểu diễn trong UML (2)

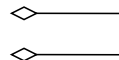
- *Điểm* được gọi là đối tượng thành phần
- *Tứ giác* là lớp chứa đối tượng thành phần
- *Điểm* được khai báo và sử dụng giống như các dữ liệu thành phần của *Tứ giác*.



2. Biểu diễn trong UML (3)

● Một số thuật ngữ liên quan:

- Kết tập
 - Aggregation (has a)
 - Composition (use a)
- Liên kết
 - Association



Chương 07. Kết tập

1. Tổng quan về kết tập
2. Biểu diễn trên UML

⇒ 3. Thực thi trên Java



3. Thực thi trên Java

```
class Diem {
    private int x, y;
    public Diem(int _x, int _y){
        x = _x; y = _y; }
    public void setX(int _x){x=_x;}
    public int getX() {return x;}//...
}
class TuGiac {
    private Diem d1, d2;
    private Diem d3, d4;
    public TuGiac(Diem _d1, Diem _d2, Diem _d3, Diem
        _d4){
        d1=_d1; d2=_d2;
        d3=_d3; d4=_d4; }
    public void setD1(Diem _d1){
        d1=_d1;}
    public Diem getD1(){return d1;}
    //...
}
```



3. Thực thi trên Java (2)

```
public class Test {
    public static void main(String arg[])
    {
        Diem d1 = new Diem(0,0);
        Diem d2 = new Diem(0,1);
        Diem d3 = new Diem (1,1);
        Diem d4 = new Diem (1,0);
        TuGiac tg1 = new TuGiac(d1, d2, d3, d4);
        TuGiac tg2 = new TuGiac();
        tg2.setD1(d1);
        // ...
    }
}
```



Ví dụ thêm

```
class Person {
    private String name;
    private Date bithday;
    public String getName() { return name; }
    ...
}
class Employee {
    private Person me;
    private double salary;
    public String getName() { return me.getName(); }
    ...
}
```



Ví dụ thêm

```
class Manager {
    private Employee me;
    private Employee assistant;
    public setAssistant(Employee e) {...}
    ...
}
...
Manager junior = new Manager();
Manager senior = new Manager();
// senior.setAssistant(junior); error
```

