School of Communication & Information Technology
Hanoi University of Science & Technology

Object-Oriented Language & Theory

Lecture 02:
Class Building & Object Usage

Nguyễn Thị Thu Trang
trangntt@soict.hust.edu.vn

1

Mục tiêu bài học

- Nêu được bản chất, vai trò của trừu tượng hóa
- Giải thích về đóng gói và che giấu thông tin
- Xây dựng lớp
  - Định nghĩa lớp, thực hiện ẩn
  - Tạo các phương thức, các trường/thuộc tính
- Tạo và sử dụng đối tượng
  - Phương thức khởi tạo
  - Khai báo và khởi tạo đối tượng
  - Sử dụng đối tượng

2

Nội dung

1. Trừu tượng hóa dữ liệu
2. Đóng gói và xây dựng lớp
3. Tạo và sử dụng đối tượng

3

1.1. Trừu tượng hóa

- Giảm thiểu và tinh lọc các chi tiết nhằm tập trung vào một số khái niệm/vấn đề quan tâm tại một thời điểm.
  - "abstraction – a concept or idea not associated with any specific instance".
  - Ví dụ: Các định nghĩa toán học
- 2 loại trừu tượng hóa
  - Trừu tượng hóa điều khiển (control abstraction)
  - Trừu tượng hóa dữ liệu (data abstraction)

4

1.1. Trừu tượng hóa (2)

- Trừu tượng hóa điều khiển: Sử dụng các chương trình con (subprogram) và các luồng điều khiển (control flow)
  - Ví dụ: a := (1 + 2) * 5
    - Nếu không có trừu tượng hóa điều khiển, LTV phải chỉ ra tất cả các thanh ghi, các bước tính toán mức nhị phân…
- Trừu tượng hóa dữ liệu: Xử lý dữ liệu theo các cách khác nhau
  - Ví dụ: Kiểu dữ liệu
    - Sự tách biệt rõ ràng giữa các thuộc tính trừu tượng của kiểu dữ liệu và các chi tiết thực thi cụ thể của kiểu dữ liệu đó.

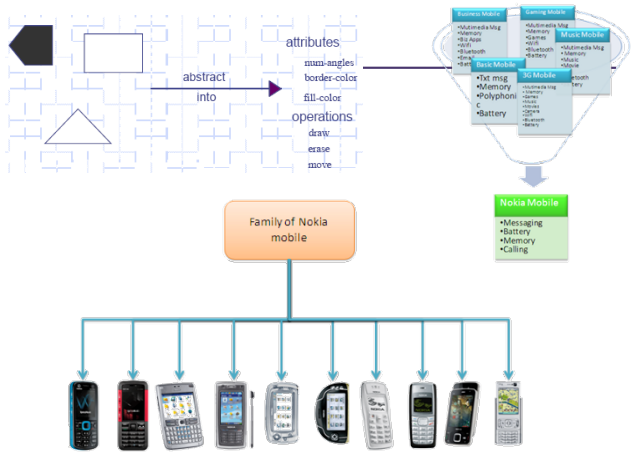
5

1.2. Trừu tượng hóa trong OOP

- Đối tượng trong thực tế phức tạp

- Cần đơn giản hóa, bỏ qua những chi tiết không cần thiết
- Chỉ "trích rút" lấy những thông tin liên quan, thông tin quan tâm, quan trọng với bài toán

6



7

7

1.2. Trừu tượng hóa trong OOP

- Any model that includes the most important, essential, or distinguishing aspects of something while suppressing or ignoring less important, immaterial, or diversionary details. The result of removing distinctions so as to emphasize commonalties (Dictionary of Object Technology, Firesmith, Eykholt, 1995).

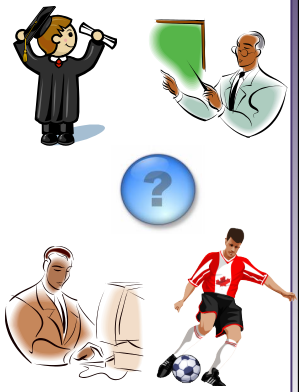
→ Cho phép quản lý các bài toán phức tạp bằng cách tập trung vào các đặc trưng quan trọng của một thực thể nhằm phân biệt nó với các loại thực thể khác.

8

8

1.2. Trừu tượng hóa trong OOP

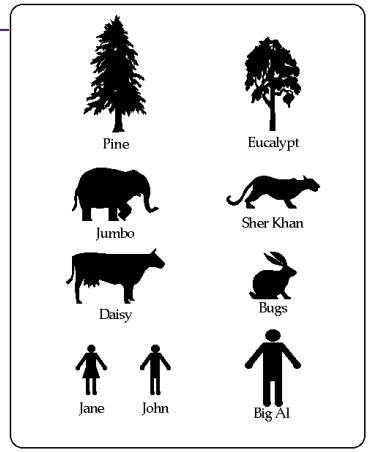
- Trừu tượng hóa là một cách nhìn hoặc cách biểu diễn một thực thể chỉ bao gồm các thuộc tính liên quan trong một ngữ cảnh nào đó
- Tập hợp các thể hiện của các thực thể thành các nhóm có chung các thuộc tính gọi là Lớp (class)



9

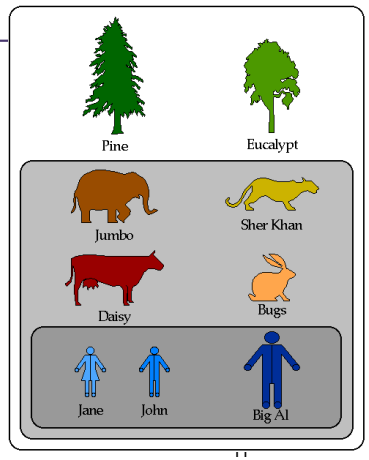
9

unclassified "things"



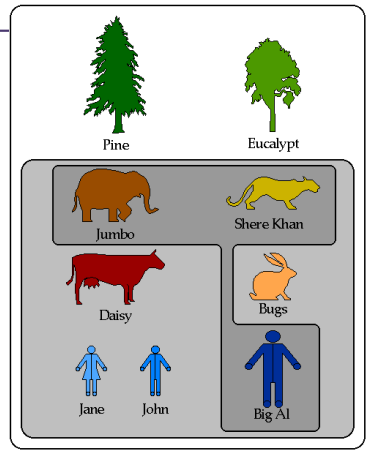
10

- organisms, mammals, humans



11

- organisms, mammals, dangerous mammals



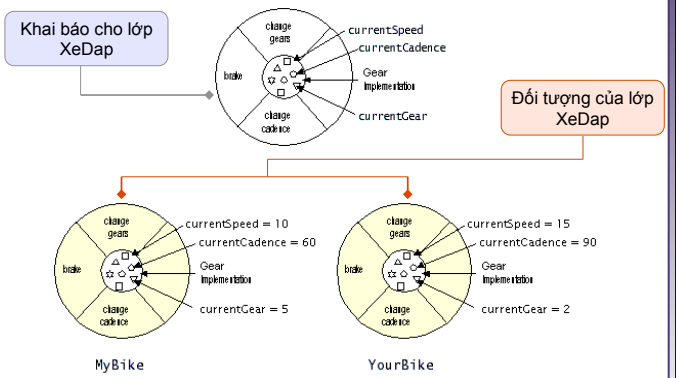
12

1.3. Lớp và Đối tượng

- Một lớp là một thiết kế (blueprint) hay mẫu (prototype) cho các đối tượng cùng kiểu
  - Ví dụ: lớp XeDap là một thiết kế chung cho nhiều đối tượng xe đạp được tạo ra
- Lớp định nghĩa các thuộc tính và các phương thức chung cho tất cả các đối tượng của cùng một loại nào đó
- Một đối tượng là một thể hiện cụ thể của một lớp
  - VD mỗi đối tượng xe đạp là một thể hiện của lớp XeDap
- Mỗi thể hiện có thể có những thuộc tính thể hiện khác nhau
  - VD một xe đạp có thể đang ở bánh răng thứ 5 trong khi một xe khác có thể là đang ở bánh răng thứ 3.

13

Ví dụ Lớp Xe đạp



14

14

1.3. Lớp và Đối tượng

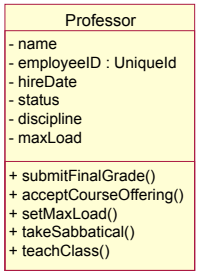
- Lớp là mô hình khái niệm, mô tả các thực thể
- Lớp như một bản mẫu, định nghĩa các thuộc tính và phương thức chung của các đối tượng
- Một lớp là sự trừu tượng hóa của một tập các đối tượng

- Đối tượng là sự vật thật, là thực thể thực sự
- Đối tượng là một thể hiện (instance) của một lớp, *dữ liệu của các đối tượng khác nhau là khác nhau*
- Mỗi đối tượng có một lớp xác định dữ liệu và hành vi của nó.

15

Biểu diễn lớp trong UML

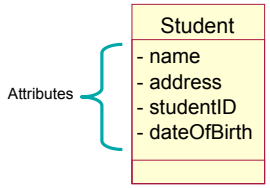
- Lớp (class) được biểu diễn bằng 1 hình chữ nhật với 3 thành phần:
  - Tên lớp
  - Dữ liệu (thuộc tính)
  - Hành vi (thao tác)



16

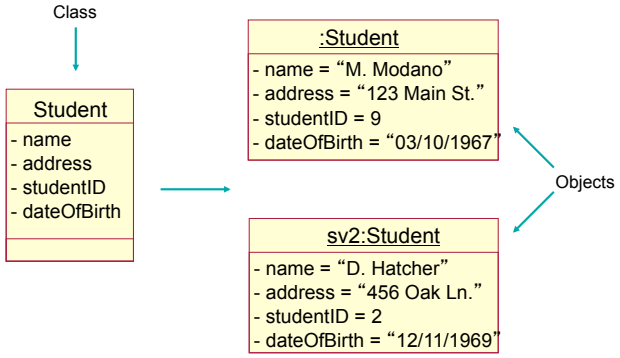
Thuộc tính (attribute) là gì?

- Một thuộc tính là một đặc tính được đặt tên của một lớp mô tả khoảng giá trị mà các thể hiện của đặc tính đó có thể chứa
  - Một lớp có thể không có thuộc tính nào hoặc có số lượng thuộc tính bất kỳ



17

Lớp và đối tượng trong UML



18

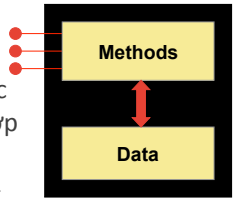
Nội dung

1. Trừu trượng hóa dữ liệu
2. Đóng gói và xây dựng lớp
3. Tạo và sử dụng đối tượng

19

2.1. Đóng gói (Encapsulation)

- 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

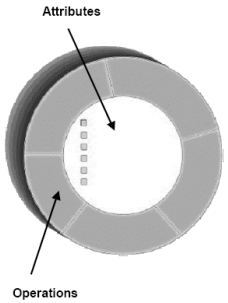


20

20

2.1. Đóng gói (2)

- Dữ liệu/thuộc tính và hành vi/phương thức được đóng gói trong một lớp → Encapsulation



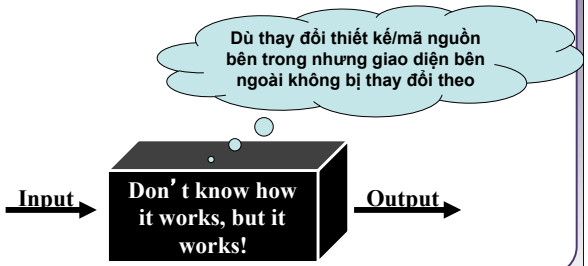
BankAccount
- owner: String - balance: double
+ debit(double): boolean +credit(double)

21

21

2.1. Đóng gói (3)

- Một đối tượng là một thực thể được đóng gói, cung cấp tập các dịch vụ nhất định
- Một đối tượng được đóng gói có thể được xem như một hộp đen – các công việc bên trong là ẩn so với client



22

2.2. Xây dựng lớp

- Thông tin cần thiết để định nghĩa một lớp
  - Tên (Name)
    - Tên lớp nên mô tả đối tượng trong thế giới thật
    - Tên lớp nên là số ít, ngắn gọn, và xác định rõ ràng cho sự trừu tượng hóa
  - Danh sách các phần tử dữ liệu
    - Các phần dữ liệu cần lấy ra khi trừu tượng hóa
  - Danh sách các thông điệp
    - Các thông điệp mà đối tượng đó có thể nhận được

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

23

2.2. Xây dựng lớp (3)

- Các lớp được nhóm lại thành package
  - Package bao gồm một tập hợp các lớp có quan hệ logic với nhau,
  - Package được coi như các thư mục, là nơi tổ chức các lớp, giúp xác định vị trí dễ dàng và sử dụng các lớp một cách phù hợp.
- Ví dụ:
  - Một số package có sẵn của Java: java.lang, javax.swing, java.io…
  - Package có thể do ta tự đặt
    - Cách nhau bằng dấu "."
    - Quy ước sử dụng ký tự thường để đặt tên package
    - Ví dụ: package oop.k52.cnpm;

24

24

2.2.1. Khai báo lớp

- Cú pháp khai báo:

```
package tenpackage;
chi_dinh_truy_cap class TenLop {
  // Than lop
}
```

  - **chi_dinh_truy_cap:**
    - **public**: Lớp có thể được truy cập từ bất cứ đâu, kể cả bên ngoài package chứa lớp đó.
    - **private**: Lớp chỉ có thể được truy cập trong phạm vi lớp đó
    - Không có (mặc định): Lớp có thể được truy cập từ bên trong package chứa lớp đó.

25

25

Ví dụ - Khai báo lớp

```
package oop.k52.cnpm;

public class Student {
  ...
}
```

26

2.2.2. Khai báo thành viên của lớp

- Các thành viên của lớp cũng có chỉ định truy cập tương tự như lớp

	public	Không có	private
Cùng lớp			
Cùng gói			
Khác gói			

27

2.2.2. Khai báo thành viên của lớp

- Các thành viên của lớp cũng có chỉ định truy cập tương tự như lớp

	public	Không có	private
Cùng lớp	Yes	Yes	Yes
Cùng gói	Yes	Yes	No
Khác gói	Yes	No	No

28

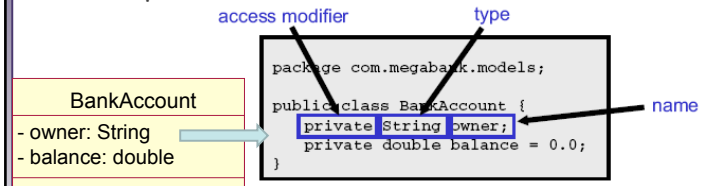
a. Thuộc tính (instance attribute)

- Các thuộc tính phải được khai báo bên trong lớp
- Mỗi đối tượng có bản sao các thuộc tính của riêng nó
  - Giá trị của một thuộc tính thuộc các đối tượng khác nhau là khác nhau

29

a. Thuộc tính

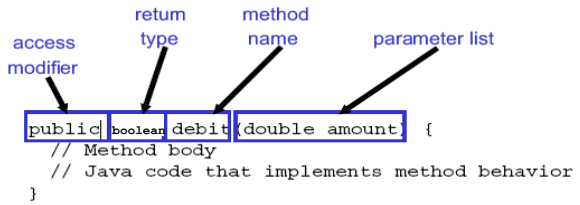
- Thuộc tính có thể được khởi tạo khi khai báo
  - Các giá trị mặc định sẽ được sử dụng nếu không được khởi tạo



30

b. Phương thức

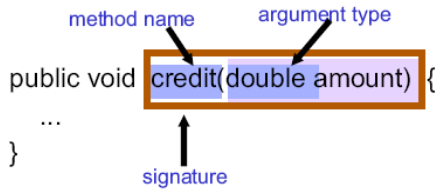
- Xác định cách một đối tượng đáp ứng lại thông điệp
- Phương thức xác định các hoạt động của lớp
- Bất kỳ phương thức nào cũng phải thuộc về một lớp nào đó



31

* Chữ ký phương thức (signature)

- Mỗi phương thức phải có một chữ ký riêng gồm:
  - Tên phương thức
  - Số lượng các tham số và kiểu của chúng



32

* Kiểu dữ liệu trả về

- Khi phương thức trả về ít nhất một giá trị hoặc một đối tượng thì bắt buộc phải có câu lệnh return để trả điều khiển cho đối tượng gọi phương thức.
- Nếu phương thức không trả về 1 giá trị nào (void) và có thể không cần câu lệnh return
- Có thể có nhiều lệnh return trong một phương thức; câu lệnh đầu tiên mà chương trình gặp sẽ được thực thi.

33

c. Thành viên hằng

- Một thuộc tính/phương thức không thể thay đổi giá trị/nội dung trong quá trình sử dụng.
- Cú pháp khai báo:

```
chi_dinh_truy_cap final
kieu_du_lieu
  TEN_HANG = gia_tri;
```

- Ví dụ:

```
final double PI = 3.141592653589793;
public final int VAL_THREE = 39;
private final int[] A = { 1, 2, 3, 4,
 5, 6 };
```

34

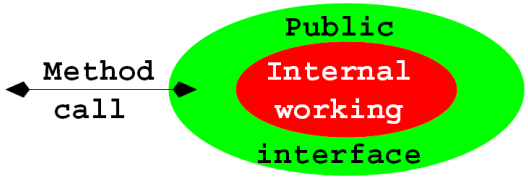
```
package com.megabank.models;
public class BankAccount {
    private String owner;
    private double balance;

    public boolean debit(double amount){
        if (amount > balance)
            return false;
        else {
            balance -= amount; return true;
        }
    }
    public void credit(double amount){
        balance += amount;
    }
}
```

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

35

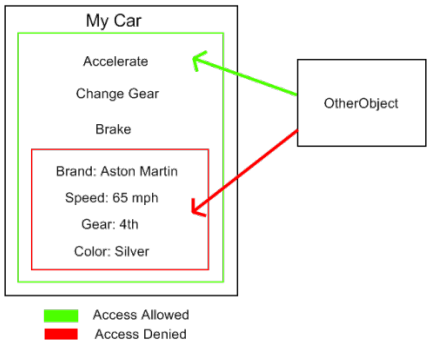
2.3. Che giấu dữ liệu (Data hiding)

- 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



36

Ví dụ - Che giấu dữ liệu



37

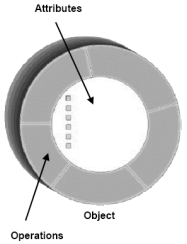
Thảo luận

- Đóng gói dữ liệu có phải là che giấu dữ liệu không?

38

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)

39

Cơ chế che giấu dữ liệu (2)

- Vì dữ liệu là riêng tư → Thông thường một lớp cung cấp các dịch vụ để truy cập và chỉnh sửa các giá trị của dữ liệu
  - 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;
}
```

40

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*

41

Bài tập 1

NhanVien
-tenNhanVien: String
-luongCoBan: double
-heSoLuong: double
+LUONG_MAX: double
+tangLuong(double):boolean
+tinhLuong(): double
+inTTin()

- Viết mã nguồn cho lớp NhanVien như trong hình bên biết:
  - Lương = Lương cơ bản * Hệ số lương
  - Phương thức inTTin() hiển thị thông tin của đối tượng NhanVien tương ứng
  - Phương thức tangLuong(double) tăng hệ số lương hiện tại lên một lượng bằng giá trị tham số double truyền vào. Nếu điều này làm cho lương của nhân viên > lương tối đa cho phép thì không cho phép thay đổi, in ra thông báo và trả về false, ngược lại trả về true.
- Viết các phương thức get và set cho các thuộc tính của lớp NhanVien.

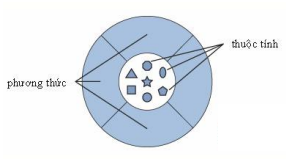
42

Nội dung

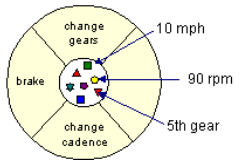
1. Trừu trượng hóa dữ liệu
2. Đóng gói và xây dựng lớp
3. Tạo và sử dụng đối tượng

43

Đối tượng



Đối tượng phần mềm



Đối tượng phần mềm **Xe Đạp**

Đối tượng (object) là một thực thể phần mềm bao bọc các **thuộc tính** và các **phương thức** liên quan.

Thuộc tính được xác định bởi giá trị cụ thể gọi là **thuộc tính thể hiện**.

Một đối tượng cụ thể được gọi là một **thể hiện**.

44

3.1. Khởi tạo dữ liệu

- Dữ liệu cần được khởi tạo trước khi sử dụng
  - Lỗi khởi tạo là một trong các lỗi phổ biến
- Với kiểu dữ liệu đơn giản, sử dụng toán tử =
- Với đối tượng → Cần dùng phương thức khởi tạo

45

Khởi tạo và hủy bỏ đối tượng

- Mỗi đối tượng khi tồn tại và hoạt động được hệ điều hành cấp phát một vùng nhớ để lưu lại các giá trị của dữ liệu thành phần
- Khi tạo ra đối tượng HĐH sẽ gán giá trị khởi tạo cho các dữ liệu thành phần
  - Phải được thực hiện tự động trước khi người lập trình có thể tác động lên đối tượng
  - Sử dụng hàm/phương thức khởi tạo
- Ngược lại khi kết thúc cần phải giải phóng hợp lý tất cả các bộ nhớ đã cấp phát cho đối tượng.
  - Java: JVM
  - C++: Hàm hủy (destructor)

46

3.2. Phương thức khởi tạo

- Là phương thức đặc biệt được gọi tự động khi tạo ra đối tượng
- Mục đích chính: Khởi tạo cho các thuộc tính của đối tượng

47

3.2. Phương thức khởi tạo (2)

- Mỗi lớp phải chứa ít nhất một constructor
  - Có nhiệm vụ tạo ra một thể hiện mới của lớp
  - Tên của constructor trùng với tên của lớp
  - Constructor không có kiểu dữ liệu trả về
- Ví dụ:

```
public BankAccount(String o, double b){
   owner = o;
   balance = b;
}
```

48

3.2. Phương thức khởi tạo (3)

- Phương thức khởi tạo **có thể dùng** chỉ định truy cập
  - **`public`**
  - **`private`**
  - Mặc định
- Một phương thức khởi tạo **không thể dùng** các từ khóa **`abstract`, `static`, `final`, `native`, `synchronized`**.
- Các phương thức khởi tạo không được xem như là *thành viên của lớp.*

49

3.2. Phương thức khởi tạo (4)

- Phương khởi tạo mặc định (default constructor)
  - Là phương thức khởi tạo **KHÔNG THAM SỐ**

```
public BankAccount(){
  owner = "noname"; balance = 100000;
}
```

  - Nếu không viết một phương khởi tạo nào trong lớp
    - JVM mới cung cấp phương thức khởi tạo mặc định
    - Phương thức khởi tạo mặc định do JVM cung cấp có chỉ định truy cập giống như lớp của nó
  - Một lớp nên có phương thức khởi tạo mặc định

50

3.3. Khai báo và khởi tạo đối tượng

- Đối tượng được tạo ra, thể hiện hóa (instantiate) từ một mẫu chung (lớp).
- Các đối tượng phải được khai báo *kiểu* của đối tượng trước khi sử dụng:
  - Kiểu của đối tượng là lớp các đối tượng
  - Ví dụ:
    - `String strName;`
    - `BankAccount acc;`

51

3.3. Khai báo và khởi tạo đối tượng (2)

- Đối tượng cần được khởi tạo trước khi sử dụng
  - Sử dụng toán tử = để gán
  - Sử dụng từ khóa **`new`** với constructor để khởi tạo đối tượng:
    - Từ khóa **`new`** dùng để tạo ra một đối tượng mới
    - Tự động gọi phương thức khởi tạo tương ứng
  - Một đối tượng được khởi tạo mặc định là **`null`**
- Đối tượng được thao tác thông qua *tham chiếu (~* con trỏ).
- Ví dụ:

```
BankAccount acc1;
acc1 = new BankAccount();
```

52

3.3. Khai báo và khởi tạo đối tượng (3)

- Có thể kết hợp vừa khai báo vào khởi tạo đối tượng
  - Cú pháp:

```
Ten_lop ten_doi_tuong = new
  Pthuc_khoi_tao(ds_tham_so);
```

  - Ví dụ:

```
BankAccount account = new
BankAccount();
```

53

3.3. Khai báo và khởi tạo đối tượng (4)

- Phương thức khởi tạo **không có giá trị trả về**, nhưng khi sử dụng với từ khóa **new** trả về một tham chiếu đến đối tượng mới

```
public BankAccount(String name) {
   setOwner(name);
}
```

Constructor definition

Constructor use

```
BankAccount account = new BankAccount("Joe Smith");
```

54

3.3. Khai báo và khởi tạo đối tượng (5)

- Mảng các đối tượng được khai báo giống như mảng dữ liệu cơ bản
- Mảng các đối tượng được khởi tạo mặc định với giá trị **null**.
- Ví dụ:

```
Employee emp1 = new Employee(123456);
Employee emp2;
emp2 = emp1;
Department dept[] = new Department[100];
Test[] t = {new Test(1),new Test(2)};
```

55

Ví dụ 1

```
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

56

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

57

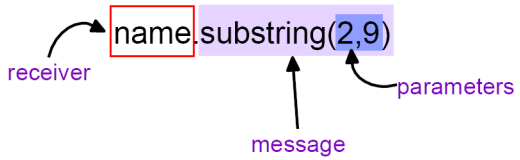
Ví dụ 3

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

58

3.4. Sử dụng đối tượng

- Đối tượng cung cấp các hoạt động phức tạp hơn các kiểu dữ liệu nguyên thủy
- Đối tượng đáp ứng lại các thông điệp
  - Toán tử "." được sử dụng để gửi một thông điệp đến một đối tượng



59

3.4. Sử dụng đối tượng (2)

- Để gọi thành viên (dữ liệu hoặc thuộc tính) của lớp hoặc đối tượng, sử dụng toán tử “.”
- Nếu gọi phương thức ngay trong lớp thì toán tử “.” không cần thiết.

```
BankAccount account = new BankAccount();
account.setOwner("Smith");
account.credit(1000.0);
System.out.println(account.getBalance());
...
```

BankAccount method

```
public void credit(double amount) {
   setBalance(getBalance() + amount);
}
```

60

```
public class BankAccount{
   private String owner;
   private double balance;
   public BankAccount(String name)
     { setOwner(name);
   }
   public void setOwner(String o){ owner = o; }
   public String getOwner(){ return owner; }
}
public class Test{
   public static void main(String args[]){
    BankAccount acc1 = new BankAccount("");
    BankAccount acc2 = new BankAccount("Hong");
    acc1.setOwner("Hoa");
    System.out.println(acc1.getOwner()
           + " "+ acc2.getOwner());
}
```

61

Tự tham chiếu – this

- Cho phép truy cập vào đối tượng hiện tại của lớp.
- Quan trọng khi hàm/phương thức thành phần thao tác trên hai hay nhiều đối tượng.
- Xóa đi sự nhập nhằng giữa một biến cục bộ, tham số với thành phần dữ liệu của lớp
- Không dùng bên trong các khối lệnh static

62

```
public class BankAccount{
   private String owner;
   private double balance;
   public BankAccount() { }
   public void setOwner(String owner){
     this.owner = owner;
   }
   public String getOwner(){ return owner; }
}
public class Test{
   public static void main(String args[]){
     BankAccount acc1 = new BankAccount();
     BankAccount acc2 = new BankAccount();
     acc1.setOwner("Hoa");
     acc2.setOwner("Hong");
     System.out.println(acc1.getOwner() + " " +
           acc2.getOwner());
}
```

63

Bài tập 2

- Viết mã nguồn cho lớp NhanVien (đã làm)
- Viết phương thức khởi tạo với các tham số cần thiết để khởi tạo cho các thuộc tính của lớp NhanVien.

NhanVien
-tenNhanVien: String -luongCoBan: double -heSoLuong: double +LUONG_MAX: double
+tangLuong(double):boolean +tinhLuong(): double +inTTin()

- Viết lớp TestNV trong đó tạo ra 2 đối tượng của lớp NhanVien, thực hiện truyền thông điệp đến các đối tượng vừa tạo để hiển thị thông tin, hiển thị lương, tăng lương...

64

Thảo luận

- So sánh Thông điệp (Message) và Phương thức (Method)?
- So sánh Thao tác (Operation) và Phương thức (Method)?
- So sánh gọi hàm (Call function) và gửi thông điệp (Send message)?

65

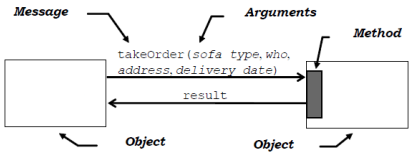
Gọi hàm vs. Gửi thông điệp

- Gọi hàm (Call function)
  - Chỉ ra chính xác đoạn mã nào sẽ được thực hiện.
  - Chỉ có duy nhất một sự thực thi của một hàm với một tên nào đó.
  - Không có hai hàm trùng tên
- Gửi thông điệp
  - Yêu cầu một dịch vụ từ một đối tượng và đối tượng sẽ quyết định cần phải làm gì
  - Các đối tượng khác nhau sẽ có các cách thực thi các thông điệp theo cách khác nhau.

66

Thông điệp vs. Phương thức

- Thông điệp
  - Được gửi từ đối tượng này đến đối tượng kia, không bao gồm đoạn mã thực sự sẽ được thực thi
- Phương thức
  - Thủ tục/hàm trong ngôn ngữ lập trình cấu trúc
  - Là sự thực thi dịch vụ được yêu cầu bởi thông điệp
  - Là đoạn mã sẽ được thực thi để đáp ứng thông điệp được gửi đến cho đối tượng



67

67