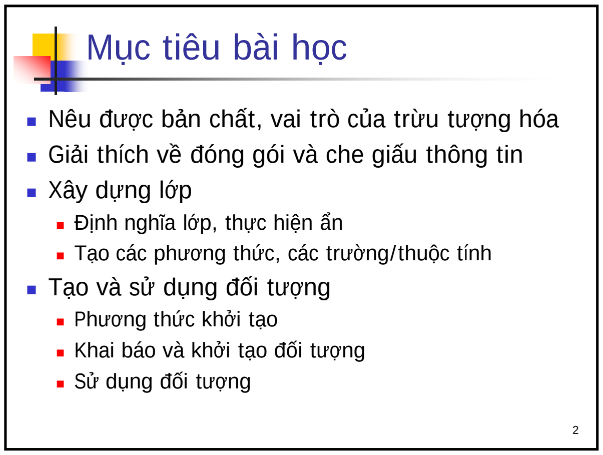
Bộ môn Công nghệ Phần mềm
Viện CNTT & TT
Trường Đại học Bách Khoa Hà Nội

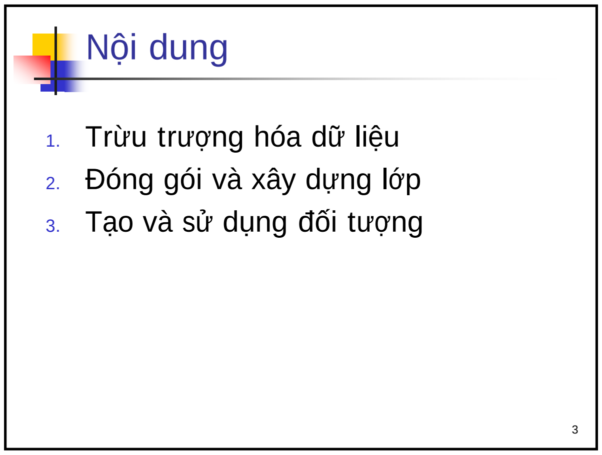
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
Bài 03. Đóng gói và xây dựng lớp, tạo và sử dụng đối tượng



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

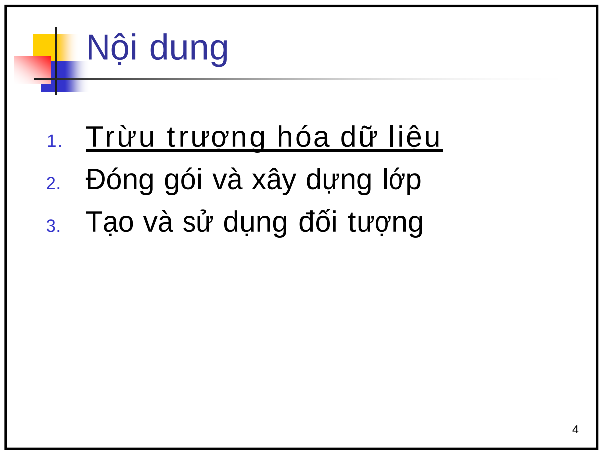
2



Nội dung

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

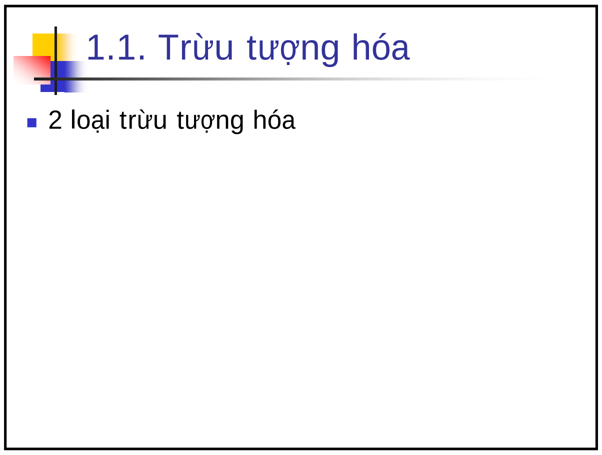
3



Nội dung

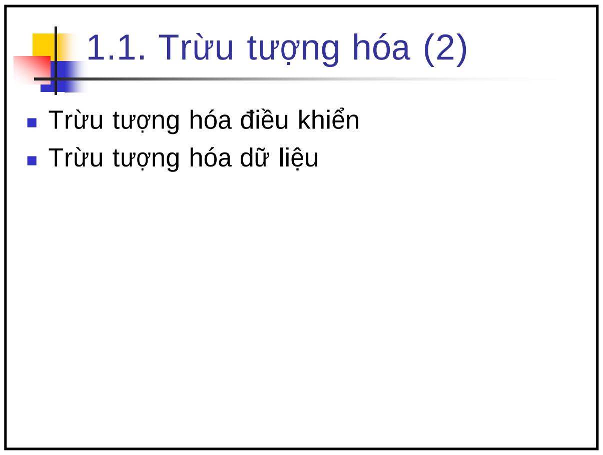
1. Trừu tượng hóa dữ liệu
2. Đóng gói và xây dựng lớp
3. Tạo và sử dụng đối tượng

4



1.1. Trừu tượng hóa

- 2 loại trừu tượng hóa



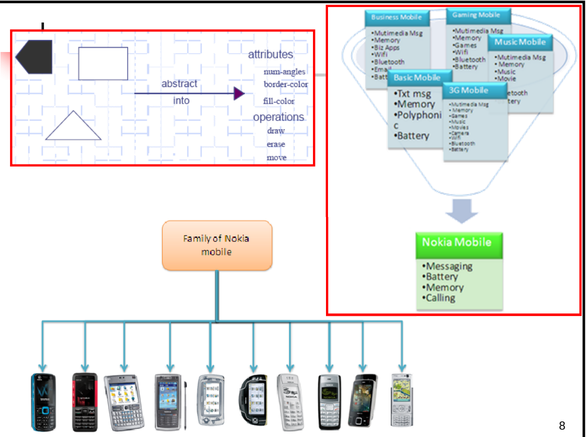
1.1. Trừu tượng hóa (2)

- Trừu tượng hóa điều khiển
- Trừu tượng hóa dữ liệu

1.2. Trừu tượng hóa dữ liệu trong OOP

- Đối tượng trong thực tế phức tạp

7



8

1.2. Trừu tượng hóa dữ liệu (3)

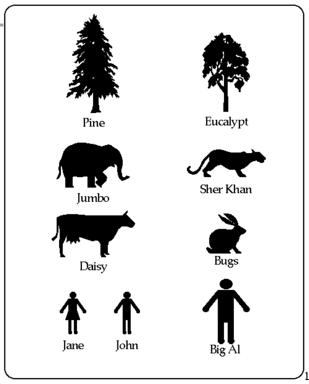
- Any model that includes the most important, essential, or distinguishing aspects of something while suppressing or ignoring less important, immaterial, or diversionary details. The result of removing distinctions so as to emphasize commonalties (*Dictionary of Object Technology*, Firesmith, Eykholt,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

9

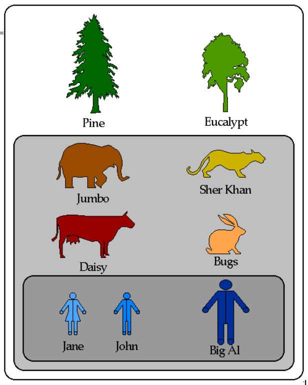
1.2. Trừu tượng hóa dữ liệu (2)

- Tập hợp các thể hiện của các thực thể thành các nhóm có chung các thuộc tính

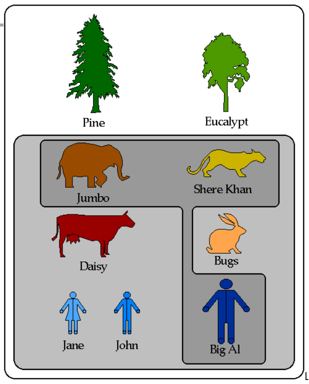
10



11



12



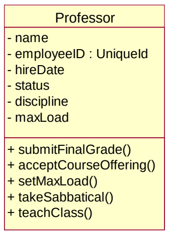
13

1.3. Lớp vs. Đối tượng

14

Biểu diễn lớp trong UML

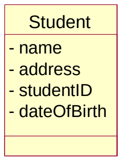
- 3 thành phần:



Professor
- name - employeeID : UniqueId - hireDate - status - discipline - maxLoad

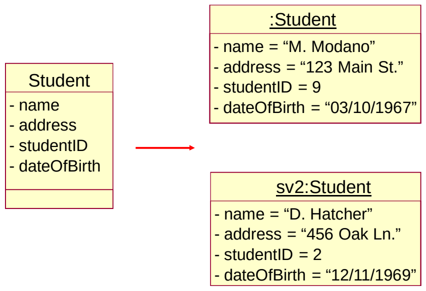
+ submitFinalGrade()
+ acceptCourseOffering()
+ setMaxLoad()
+ takeSabbatical()
+ teachClass()

Thuộc tính (attribute) là gì?



Student
- name - address - studentID - dateOfBirth

Lớp và đối tượng trong UML



Student
- name - address - studentID - dateOfBirth

:Student
- name = “M. Modano” - address = “123 Main St.” - studentID = 9 - dateOfBirth = “03/10/1967”

sv2:Student
- name = “D. Hatcher” - address = “456 Oak Ln.” - studentID = 2 - dateOfBirth = “12/11/1969”

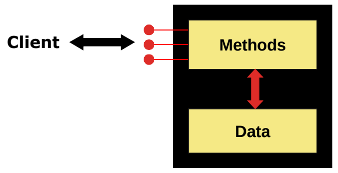
Nội dung

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

18

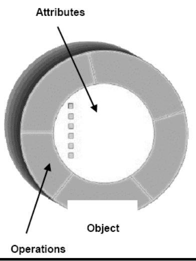
2.1. Đóng gói (Encapsulation)

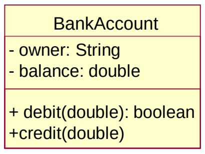
- Một đối tượng có hai khung nhìn:



19

2.1. Đóng gói (2)

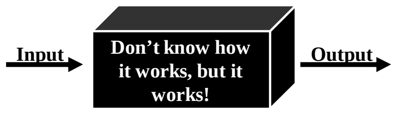




BankAccount
- owner: String - balance: double
+ debit(double): boolean +credit(double)

20

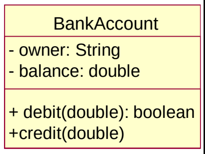
2.1. Đóng gói (3)



21

2.2. Xây dựng lớp

- Thông tin cần thiết để định nghĩa một lớp

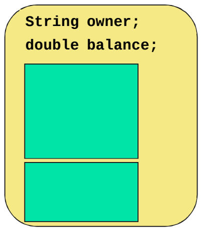


BankAccount
- owner: String - balance: double
+ debit(double): boolean +credit(double)

22

2.2. Xây dựng lớp (2)

- Lớp đóng gói các thành viên (member)



```
String owner;
double balance;
```

23

2.2. Xây dựng lớp (3)

- Các lớp được nhóm lại thành package

24

2.2.1. Khai báo lớp

- Cú pháp khai báo:

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

Ví dụ - Khai báo lớp

```
package oop.k52.cnpm;

public class Student {
   …
}
```

2.2.2. Khai báo thành viên của lớp

	public	Không có	private
Cùng lớp			
Cùng gói			
Khác gói			

Ví dụ: private

```
class Student{
   private String name;
   public String getName() {
      return this.name;
   }
   public void setName(String name)
   {
      this.name = name;
   }
}
```

Ví dụ: private (2)

```
class Student{
   private String name;
   public String getName() {
      return this.name;
   }
   public void setName(String name)
    {
      this.name = name;
    }
}
class Manager{
    private Student[] students;
    public initianize()
    {
       students[0] = new Students();
       //student.name = "Hung"; error
       student.setName("Hung");
    }
}
```

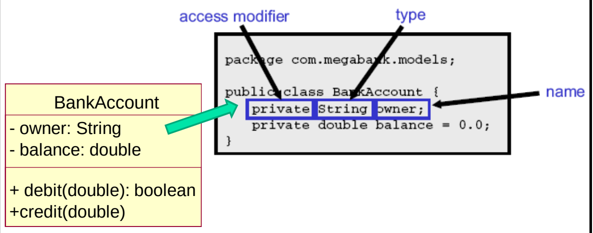
a. Thuộc tính

- Các thuộc tính phải được khai báo bên trong lớp

Student
- name - address - studentID - dateOfBirth

a. Thuộc tính (2)

- Thuộc tính có thể được khởi tạo khi khai báo

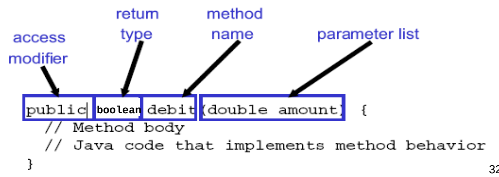


```
package com.megabank.models;

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

BankAccount
- owner: String
- balance: double

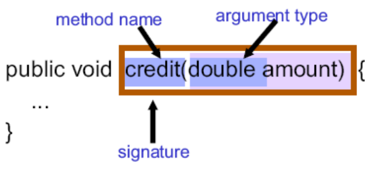
+ debit(double): boolean
+credit(double)

b. Phương thức



```
public boolean debit(double amount)  {
   // Method body
   // Java code that implements method behavior
}
```

* Chữ ký phương thức (signature)



```
public void credit(double amount) {
    ...
}
```

* Kiểu dữ liệu trả về

- Lệnh return

Ví dụ

```
public Boolean checkOdd(int i)
{
  if (i %2 ==0)
      return true;
  else
      return false;
}

public Boolean checkOdd(int i)
{
  return true;
  return false;
}
```

c. Thành viên hằng

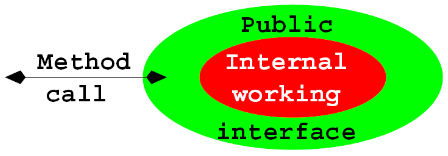
- Ví dụ:

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

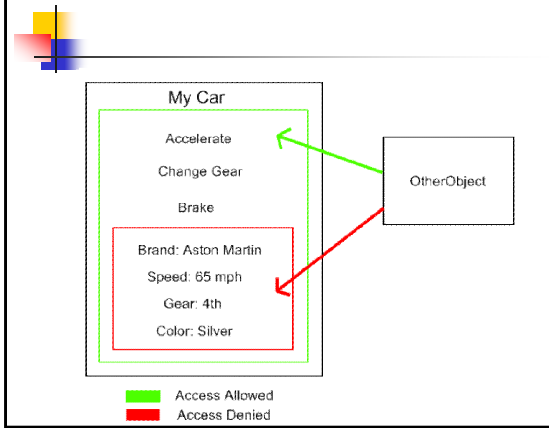
```
package com.megabank.models;
public class BankAccount {
    private String owner;
    private double balance;

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

37

2.3. Che giấu dữ liệu (Data hiding)

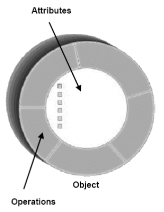


38



39

Cơ chế che giấu dữ liệu



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

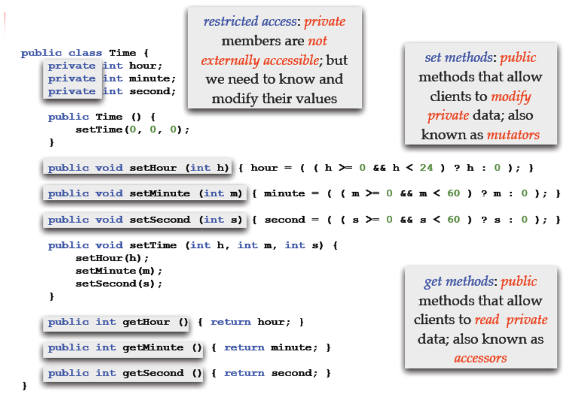
Cơ chế che giấu dữ liệu (2)

- Accessor (getter):
- Mutator (setter):

```
package com.megabank.models;

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

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

41



```
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; }
}
```

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*

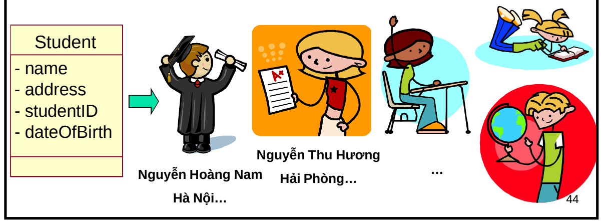
get methods: *public* methods that allow clients to *read private* data; also known as *accessors*

Nội dung

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

43

3.1. Khởi tạo dữ liệu



44

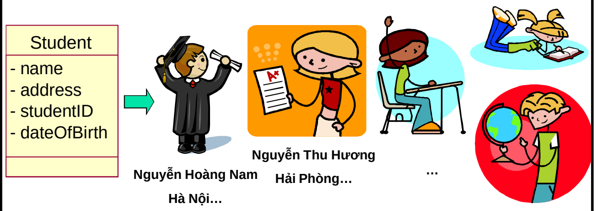
Khởi tạo và hủy bỏ đối tượng

- Tạo
- Hủy

45

3.2. Phương thức khởi tạo

- Mục đích chính?



46

3.2. Phương thức khởi tạo (2)

- Ví dụ:

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

47

3.2. Phương thức khởi tạo (3)

- Các chỉ định truy cập có thể dùng?
- Được xem như là thành viên của lớp?

48

3.2. Phương thức khởi tạo (4)

- Phương khởi tạo mặc định (default constructor)

3.3. Khai báo và khởi tạo đối tượng

- Đối tượng được tạo ra, thể hiện hóa (instantiate) từ một mẫu chung (lớp).

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

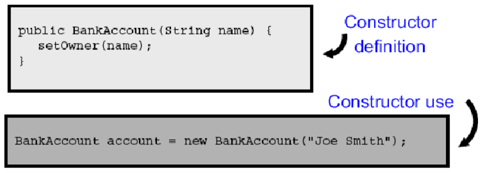
- Đối tượng cần được khởi tạo trước khi sử dụng

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

- Ví dụ:

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

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



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

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

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

- Ví dụ:

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

Ví dụ 1

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

55

Ví dụ 2

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

56

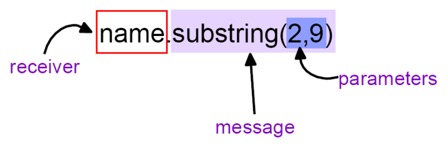
Ví dụ 3

```
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();
    BankAccount account2 = new BankAccount(“Hoang”);
   }
}
```

57

3.4. Sử dụng đối tượng

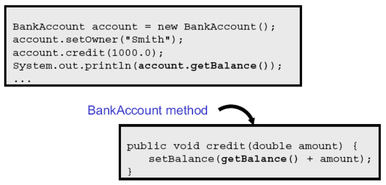
- Đối tượng cung cấp các hoạt động phức tạp hơn các kiểu dữ liệu nguyên thủy



58

3.4. Sử dụng đối tượng (2)

- Toán tử “.”



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

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

59

```
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());
}
```

60

Tự tham chiếu – this

- Cho phép truy cập vào đối tượng hiện tại của lớp.

```
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());
}
```