Lập trình Java cơ bản

Cao Đức Thông - Trần Minh Tuấn
cdthong@ifi.edu.vn, tmtuan@ifi.edu.vn

Bài 2. OOP trong Java

- Các phương pháp lập trình
- Giới thiệu về OOP
- Kế thừa (Inheritance)
- Đa hình (Polymorphism)
- Giao tiếp (Interface)
- Lớp trừu tượng (Abstract)
- Gói (Packages)
- Java vs C++
- Bài tập

Các phương pháp lập trình

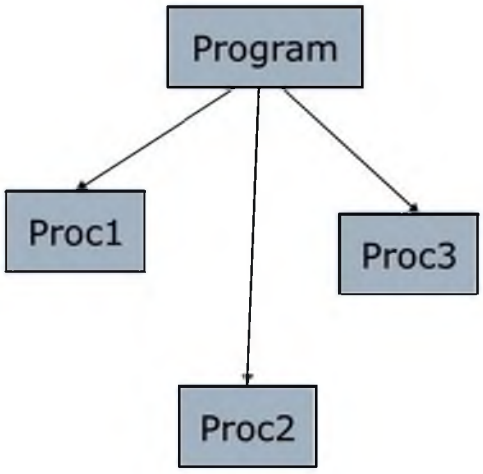
- Step-by-Step Programming
  - Lập trình từng bước
  - Machine Language, Assembly Language

```
1101011100111101
1010000111011101
0110111011001010
1000100100101011
...
```

```
MOV BX, CS
SUB BX, 10h
JNE label1
LOOP label2
...
```

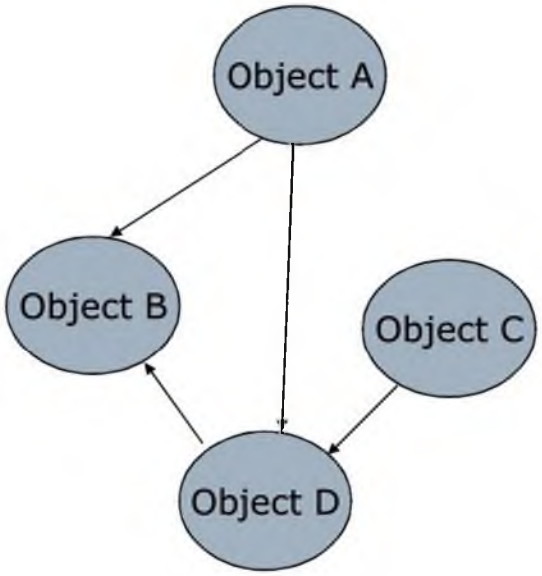
Các phương pháp lập trình

- Procedural Programming
  - Thiết kế top-down. Chương trình được chia thành các hàm, thủ tục.
  - Pascal, C
  - Hạn chế: dùng dữ liệu toàn cục, khó phát triển, khó mô tả các thực thể trong thực tế



Các phương pháp lập trình

- Object-Oriented Programming
  - Chương trình bao gồm các đối tượng. Các đối tượng tương tác với nhau thông qua các phương thức của chúng
  - Dễ mô tả các thực thể
  - Dễ phát triển, mở rộng chương trình



Lập trình hướng đối tượng-OOP

- Lớp và đối tượng
  - Dữ liệu và các thao tác trên dữ liệu được kết hợp trong cùng một đối tượng (object)
  - Lớp (class) định nghĩa các tính chất của một tập hợp các đối tượng cùng kiểu
  - Đối tượng là các thể hiện (instances) của lớp
- Đặc điểm của OOP
  - Tính đóng gói (Encapsulation)
  - Tính kế thừa (Inheritance)
  - Tính đa hình (Polymorphism)

Ví dụ về OOP trong Java

- Lớp Time và TimeTest nằm trong cùng thư mục
  - Lớp Time nằm trong file Time.java. Nó chứa các định nghĩa về thời gian.
  - Lớp TimeTest nằm trong file TimeTest.java. Nó dùng để kiểm tra lớp Time. Lớp TimeTest chứa hàm main.
- Khi chạy chỉ cần gõ:
  - *javac TimeTest.java*
  - *java TimeTest*
  - Java sẽ tự động tìm và dịch file Time.java

Lớp Time

```
// File Time.java
import java.text.DecimalFormat;
public class Time
{
      private int hour;      // 0 - 23
      private int minute;    // 0 - 59
      private int second;    // 0 - 59
      // Cau tu
      public Time() { setTime( 0, 0, 0 ); }
      // Ham thiet lap du lieu
      public void setTime( int h, int m, int s )
      {
         hour = ( ( h >= 0 && h < 24 ) ? h : 0 );
         minute = ( ( m >= 0 && m < 60 ) ? m : 0 );
         second = ( ( s >= 0 && s < 60 ) ? s : 0 );
      }
```

Lớp Time

```
   // Chuyen sang khuon dang thoi gian quoc te
   public String toUniversalString()
   {
      DecimalFormat twoDigits = new DecimalFormat( "00" );
      return twoDigits.format( hour ) + ":" +
    twoDigits.format( minute ) + ":" + twoDigits.format( second );
   }

   // Chuyen sang khuon dang thoi gian thong thuong
   public String toStandardString()
   {
      DecimalFormat twoDigits = new DecimalFormat( "00" );
      return ( (hour == 12 || hour == 0) ? 12 : hour % 12 ) + ":"
+ twoDigits.format( minute ) + ":" + twoDigits.format( second )
+ ( hour < 12 ? " AM" : " PM" );
   }
} // Ket thuc lop Time
```

Lớp Time

- Dữ liệu
  - Mỗi dữ liệu phải có một phạm vi nhất định
- Phương thức
  - Cấu tử có tên trùng với tên lớp
  - Cấu tử không có kiểu trả về
  - Có thể có nhiều cấu tử (overloading)
  - Mỗi phương thức phải có một phạm vi nhất định

Lớp TimeTest

```
// File TimeTest.java
import javax.swing.JOptionPane;
public class TimeTest
{
   public static void main( String args[ ] )
   {
      Time time = new Time();  // Tao mot doi tuong kieu Time
      // Lay thoi gian
      String output = "The initial universal time is: "
                      + time.toUniversalString()
                      + "\nThe initial standard time is: “
                      + time.toStandardString();

      // Dat lai thoi gian
      time.setTime( 13, 27, 6 );
      output += "\n\nUniversal time after setTime is: "
                + time.toUniversalString()
                + "\nStandard time after setTime is: "
                + time.toStandardString();
```

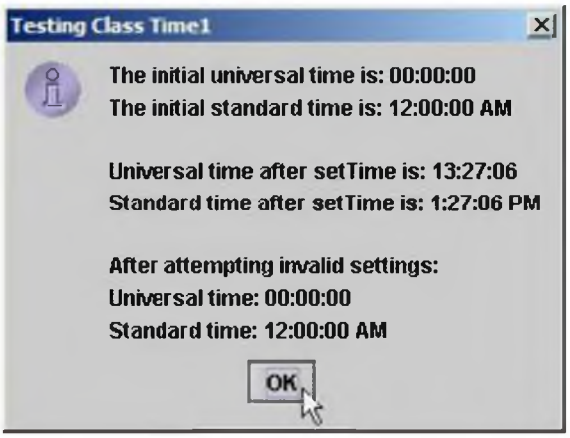
Lớp TimeTest

```
        // Dat lai thoi gian
        time.setTime( 99, 99, 99 );
        output += "\n\nAfter attempting invalid settings: "
           + "\nUniversal time: " + time.toUniversalString()
           + "\nStandard time: " + time.toStandardString();

        JOptionPane.showMessageDialog( null, output,
         "Testing Class Time1", JOptionPane.INFORMATION_MESSAGE );
        System.exit( 0 );
    } // Ket thuc ham main
} // Ket thuc lop TimeTest
```

Chạy TimeTest

- javac TimeTest.java
- java TimeTest



Phạm vi truy cập

- private
  - Chỉ truy cập được từ trong lớp khai báo.
- protected
  - Truy cập được từ trong lớp khai báo, lớp con của của lớp khai báo và các lớp cùng gói với lớp khai báo.
- public
  - Truy cập được từ mọi nơi.
- Mặc định
  - Truy cập được từ trong lớp khai báo và các lớp cùng gói với lớp khai báo.

Từ khoá this

- Từ khoá this chỉ đối tượng hiện thời

```
public class Time
{
      private int hour;      // 0 - 23
      private int minute;    // 0 - 59
      private int second;    // 0 - 59
      // Viet lai ham setTime
      public void setTime( int hour, int minute, int second )
      {
        this.hour = ( ( hour >= 0 && hour < 24 ) ? hour : 0 );
        this.minute = ( ( minute >= 0 && minute < 60 ) ? minute : 0 );
        this.second = ( ( second >= 0 && second < 60 ) ? second : 0 );
      }
```

Nạp chồng hàm khởi tạo

```
// Ham khoi tao khong tham so
public Time()        { this( 0, 0, 0 ); }
// Ham khoi tao mot tham so
public Time( int h ) { this( h, 0, 0 );   }
// Ham khoi tao hai tham so
public Time( int h, int m )        { this( h, m, 0 );  }
// Ham khoi tao ba tham so
public Time( int h, int m, int s ) {  setTime( h, m, s ); }
// Ham sao chep
public Time( Time time )
          { this( time.hour, time.minute, time.second ); }
// Su dung cac cau tu
Time t1 = new Time();              // 00:00:00
Time t2 = new Time( 2 );           // 02:00:00
Time t3 = new Time( 21, 34 );       // 21:34:00
Time t4 = new Time( 12, 25, 42 );  // 12:25:42
Time t5 = new Time( 27, 74, 99 );  // 00:00:00
Time t6 = new Time( t4 );          // 12:25:42
```

Một số từ khoá

- Từ khoá final
  - Áp dụng cho lớp, phương thức, biến.
  - Lớp final: là lớp không thể có lớp con

    public final class NoChild {...}
  - Biến final: là biến không thể thay đổi khi đã gán giá trị

    private final int MAX = 100;
  - Phương thức final: là phương thức không thể nạp chồng

    public final void NoOverride();

Một số từ khoá

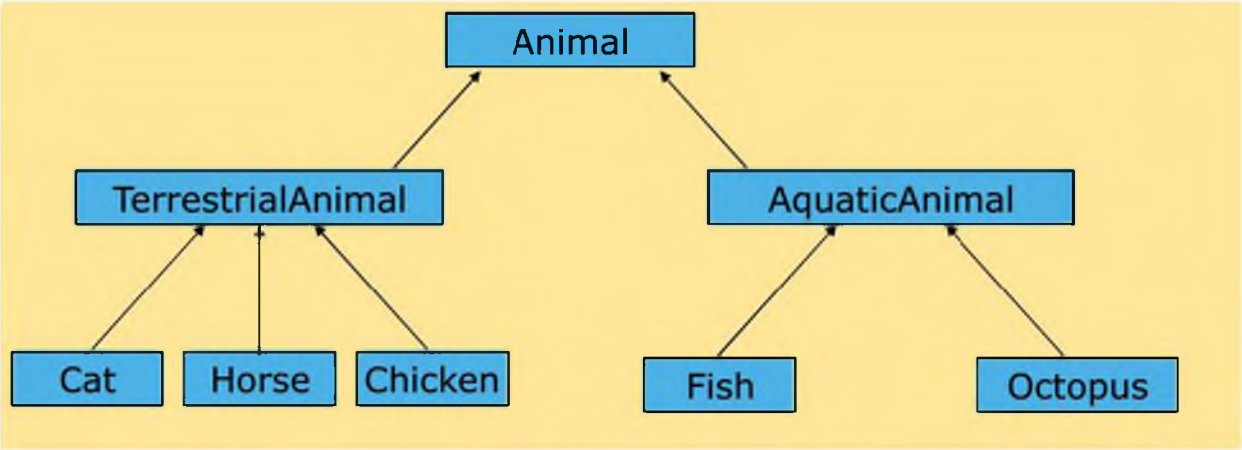
- Từ khoá static
  - Được dùng với phương thức và biến.
  - Biến static: là biến chung cho mọi đối tượng của lớp, nó được truy cập qua đối tượng của lớp hoặc qua tên lớp.

    private static char TAB = '\t';
  - Phương thức static: là phương thức chỉ được phép truy cập tới các biến static của lớp, nó có thể gọi ngay cả khi chưa có đối tượng nào của lớp.
  - public static void Welcome() {...}

Kế thừa (Inheritance)

- Kế thừa là việc xây dựng lớp mới dựa trên lớp đã có sẵn
  - Lớp đã có sẵn là lớp cha: superclass
  - Lớp mới là lớp con: subclass
- Ví dụ:
  - Hình vuông kế thừa từ hình chữ nhật
  - Con ếch kế thừa từ loài động vật dưới nước
- Chú ý:
  - Tính kế thừa thể hiện quan hệ "is a", khác với quan hệ "has a" (composition)
  - Composition: một đối tượng chứa các đối tượng thuộc lớp khác. Ví dụ: ôtô có các bánh xe

Cây kế thừa


Animal
TerrestrialAnimal
AquaticAnimal
Cat
Horse
Chicken
Fish
Octopus

Ví dụ về kế thừa trong Java

- Xây dựng lớp Hình cầu (Sphere) kế thừa từ lớp Hình tròn (Circle)

```
// File Circle.java
public class Circle
{
        protected int radius;  // radius co the duoc truy cap tu Sphere
        public Circle( int radius )
        {
            this.radius = radius;
        }
        public int getRadius()       { return radius; }
        public double getArea()    { return Math.PI*radius*radius; }
        public double getCircum() { return 2*Math.PI*radius; }
}
```

Ví dụ về kế thừa trong Java

```
// File Sphere.java
public class Sphere extends Circle
{
        public Sphere( int radius )
        {
            super(radius); // Goi cau tu cua lop cha
        }
        // Ham tinh dien tich bet mat. Nap chong ham getArea()
        public  double getArea()
        { return 4*Math.PI*radius*radius; }

        // Ham tinh the tich
        public double getVolume()
        { return (4.0d/3)*Math.PI*radius*radius*radius; }
}
```

Sphere kế thừa từ Circle

Ví dụ về kế thừa trong Java

```
// File InheritanceTest.java
public class InheritanceTest
{
   public static void main(String[] args)
   {
      Circle circle = new Circle(5);
      Sphere sphere = new Sphere(2);
      System.out.println("Radius of circle: " + circle.getRadius());
      System.out.println("Radius of sphere: " + sphere.getRadius());
      System.out.println("Area of circle: " + circle.getArea());
      System.out.println("Area of sphere: " + sphere.getArea());
      System.out.println("Volume of sphere: " + sphere.getVolume());
   }
}
```

Từ khoá super

- Từ khoá super chỉ đối tượng của lớp cha
  - super(radius): Gọi cấu tử của lớp cha
- Question: Thêm phương thức tính diện tích mặt cắt qua tâm hình cầu ?
- Answer: Đó chính là diện tích hình tròn

```
// Them phuong thuc nay trong lop Sphere
public double getAreaCut() { return super.getArea(); }
// ...
// Ham main, goi Tinh dien tich mat cat
System.out.println("Area cut of sphere: " + sphere.getAreaCut());
// ...
```

Đa hình (Polymorphism)

- Ví dụ:

```
class A
{
   public void method()
    { System.out.println("method of A"); }
}

class B extends A
{
    public void method()
     {  System.out.println("method of B"); }
}

class C extends A
{
   public void method()
     {  System.out.println("method of C"); }
}
```

```
// Câu lệnh trong main
A a = new A();
a.method();

a = new B();
a.method();

C c = new C();
a = c;
a.method();

// Kết quả màn hình
method of A
method of B
method of C
```

Đa hình (Polymorphism)

- Tính đa hình thể hiện qua việc: cùng một phương thức nhưng có nội dung thực hiện khác nhau trên các đối tượng khác nhau.
- Phương thức gọi được xác định thông qua đối tượng được tham chiếu, không thông qua kiểu khai báo của tham chiếu.
- Trong Java, các phương thức luôn mang tính đa hình.

Lớp trừu tượng (abstract)

- Lớp trừu tượng chỉ được dùng làm lớp cha cho các lớp khác, nó không có các thể hiện (instance).
- Lớp trừu tượng định nghĩa các thuộc tính chung cho các lớp con của nó.
- Ví dụ có thể thiết kế lớp Hình tròn, Hình vuông... kế thừa từ lớp trừu tượng Hình. Lớp Hình có thuộc tính là tên hình, các phương thức tính diện tích, chu vi...

Lớp trừu tượng

- Lớp trừu tượng (abstract) thường có ít nhất một phương thức trừu tượng, là phương thức không có cài đặt.

  ```
  public abstract void draw();
  ```

- Khai báo lớp trừu tượng

  ```
  public abstract class ClassName {…}
  ```

- Các lớp con của một lớp cha trừu tượng phải cài đặt tất cả các phương thức trừu tượng. Nếu không nó cũng sẽ trở thành lớp trừu tượng.
- Không thể tạo các đối tượng của một lớp trừu tượng nhưng có thể khai báo biến thuộc kiểu lớp trừu tượng để tham chiếu đến các đối tượng thuộc lớp con của nó.

Lớp trừu tượng

```
public abstract class Shape
{
        static final double PI = 3.14159;
        public abstract double getArea();
        public abstract double getVolume();
}
class Circle extends Shape
{
        double radius;
        public double getArea()    { return PI*radius*radius; }
        public double getVolume() { return 0; }
}
class Cube extends Shape
{
        double a, b, c;
        public double getArea()    { return 2*(a*b+b*c+c*a); }
        public double getVolume() { return a*b*c; }
}
```

Giao tiếp (interface)

- Giao diện chỉ ra các "tính chất" mà một đối tượng có thể có, trong một "ngữ cảnh" nào đó.
  - Một người có thể khi ở nhà là một người con, ở trường là một sinh viên, ở lớp là một người bạn.
- Giao diện trong Java có thể được dùng để thể hiện sự đa kế thừa như trong C++.

Giao tiếp (interface)

- Khai báo giao tiếp

  interface Name {...}
- Một giao tiếp thường chỉ chứa các hằng static và các phương thức public chưa cài đặt.
- Một giao tiếp có thể thừa kế một giao tiếp khác.
- Một lớp có thể cài đặt (implements) một hay nhiều giao tiếp nhưng chỉ có thể thừa kế (extends) từ một lớp.

Giao tiếp (interface)

```
interface Drawable
{
    public void draw();
}

public abstract class Shape {...}

public class Circle extends Shape implements Drawable
{
    ...
    public void draw() {...}
}
```

Gói (package)

- Package cho phép nhóm một tập hợp các lớp hoặc các giao tiếp có quan hệ với nhau để dễ dàng quản lý, bảo trì, phân phối…
- Ví dụ: Tạo package Transport chứa các lớp về phương tiện đi lại: Car, Moto, Boat...
  - Đặt các lớp vào cùng thư mục Transport
  - Khai báo ở đầu mỗi lớp dòng

    package Transport;

Gói (package)

- Sử dụng package – cách 1: Khai báo import
  - Ví dụ 1: import Transport.Car;
  - Ví dụ 2: import Transport.*;
  - Ví dụ 3: import java.awt.Point;
  - ...
  - Point p = new Point(1,2);
- Sử dụng package – cách 2: Sử dụng trực tiếp
  - java.awt.Point p = new java.awt.Point(1,2);
  - javax.swing.JOptionPane.showMessageDialog(...);
- Chú ý:
  - Chỉ có thể truy cập được đến các lớp public trong các package
  - Package java.lang.* được tự động import vào mọi chương trình

Java vs C++

1. Một chương trình Java chạy chậm hơn so với một chương trình C tương ứng khoảng 20 lần.
2. Java không có kiểu liệt kê (enum), kiểu cấu trúc (struct) hay hợp (union), nó chỉ có class. Mọi biến hay hàm của Java đều nằm trong một class nào đó.
3. Java không có toán tử phạm vi (scope) ::
4. Cỡ của các kiểu dữ liệu nguyên thuỷ (primitive) trong Java không phụ thuộc vào máy. Đặc biệt kiểu char có cỡ 16 bit (Unicode).
5. Biểu thức logic trong Java chỉ nhận giá trị boolean.
6. Trong Java, mọi biến không thuộc kiểu nguyên thuỷ đều phải tạo ra bằng từ khoá new.
7. Java không có tiền xử lý (preprocessor)

Java vs C++

8. Java không có con trỏ.
9. Java không có huỷ tử (destructor), nó chỉ có phương thức finalize() được gọi bởi Garbage Collector.
10. Java không hỗ trợ đối số mặc định.
11. Java chỉ hỗ trợ kế thừa đơn, tất cả mọi lớp đều mặc định kế thừa từ lớp Object.
12. Java không có template.
13. Java không có từ khoá virtual. Tất cả mọi phương thức non-static đều luôn dùng dynamic binding.
14. Java không có quá tải toán tử (operator overloading).
15. Đối số của hàm luôn được xử lý như đối số giá trị (value argument)

Bài tập

1. Cài đặt lại các nội dung lý thuyết đã học.
2. Xây dựng lớp Candidate (Thí sinh) gồm các thuộc tính: mã, tên, ngày tháng năm sinh, điểm thi Toán, Văn, Anh và các phương thức cần thiết.
3. Xây dựng lớp TestCandidate để kiểm tra lớp trên:
   - Nhập vào 10 thí sinh
   - In ra thông tin về các thí sinh có tổng điểm lớn hơn 15

Bài tập

4. Xây dựng lớp Point có dữ liệu là 2 toạ độ x,y và các phương thức cần thiết. Thêm phương thức toString để trả về một String có dạng "[x, y]".
5. Xây dựng lớp Rect có dữ liệu là điểm trên trái và dưới phải. Quá tải các cấu tử để cung cấp nhiều cách khởi tạo. Thêm các phương thức: Kiểm tra 1 điểm có nằm trong hình chữ nhật hay không. Tính giao của 2 hình chữ nhật.

Bài tập

6. Xây dựng lớp TestRect: Sinh ngẫu nhiên 10 hình chữ nhật và tính giao đôi một của chúng.
7. Xây dựng lớp Triangular có 3 phương thức static để tính diện tích tam giác theo 3 cách khác nhau: cạnh đáy và chiều cao, công thức Heron, độ dài 2 cạnh và góc.

Bài tập

8. Xây dựng lớp Employee có name và phương thức trừu tượng là earnings(). Xây dựng lớp Boss kế thừa từ Employee có cách tính lương là một khoản cố định hàng tháng. Xây dựng lớp PieceWorker có cách tính lương dựa trên số sản phẩm làm được, lương một sản phẩm là $ 0.5. Xây dựng lớp CommissionWorker có cách tính lương là một khoản cố định + tiền hoa hồng trên số sản phẩm bán được, mỗi sản phẩm được $ 0.1 hoa hồng.