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

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài 06. Một số kỹ thuật trong kế thừa

Mục tiêu của bài học

- Trình bày nguyên lý định nghĩa lại trong kế thừa
- Đơn kế thừa và đa kế thừa
- Giao diện và lớp trừu tượng
- Sử dụng các vấn đề trên với ngôn ngữ lập trình Java.

Nội dung

1. Định nghĩa lại (Redefine/Overiding)
2. Lớp trừu tượng (Abstract class)
3. Đơn kế thừa và đa kế thừa
4. Giao diện (Interface)

Nội dung

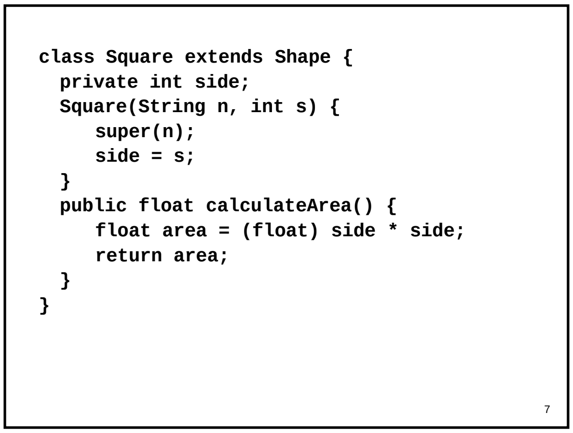
1. Định nghĩa lại (Redefine/Overriding)
2. Lớp trừu tượng (Abstract class)
3. Đơn kế thừa và đa kế thừa
4. Giao diện (Interface)

1. Định nghĩa lại hay ghi đè

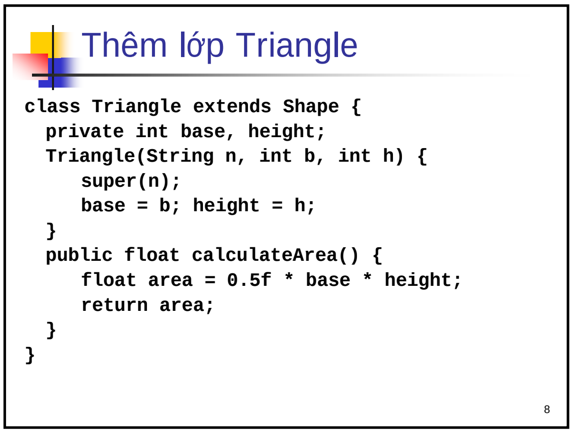
- Lớp con có thể định nghĩa phương thức trùng tên với phương thức trong lớp cha:

```
class Shape {
  protected String name;
  Shape(String n) { name = n; }
  public String getName() { return name; }
  public float calculateArea() { return 0.0f; }
}
class Circle extends Shape {
  private int radius;
  Circle(String n, int r){
     super(n);
     radius = r;
  }

  public float calculateArea() {
     float area = (float) (3.14 * radius *
  radius);
     return area;
  }
}
```

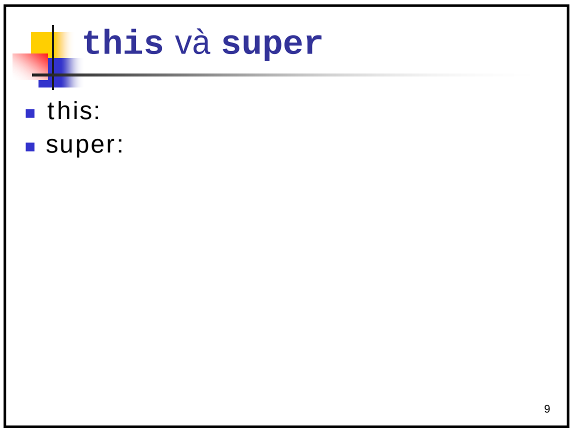


```
class Square extends Shape {
  private int side;
  Square(String n, int s) {
     super(n);
     side = s;
  }
  public float calculateArea() {
     float area = (float) side * side;
     return area;
  }
}
```



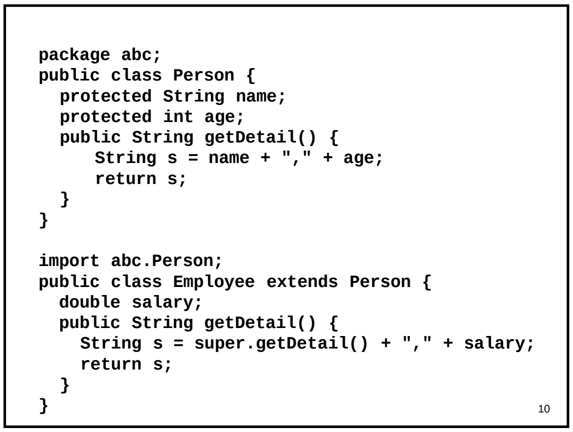
Thêm lớp Triangle

```
class Triangle extends Shape {
  private int base, height;
  Triangle(String n, int b, int h) {
     super(n);
     base = b; height = h;
  }
  public float calculateArea() {
     float area = 0.5f * base * height;
     return area;
  }
}
```



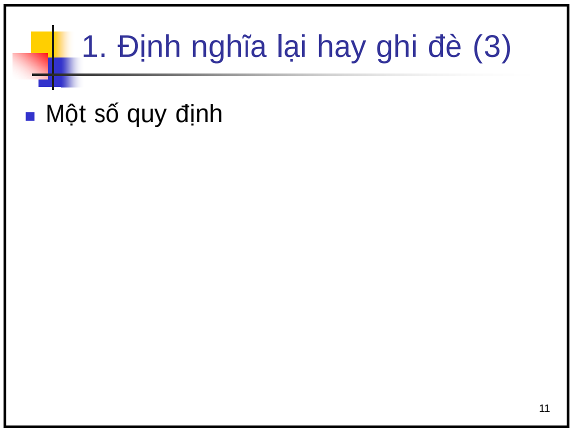
this và super

- this:
- super:



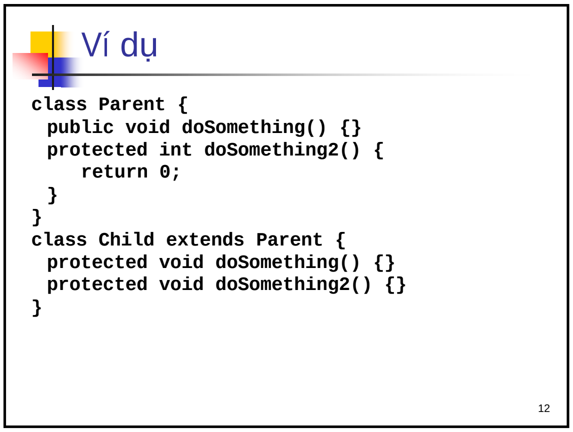
```
package abc;
public class Person {
  protected String name;
  protected int age;
  public String getDetail() {
     String s = name + "," + age;
     return s;
  }
}

import abc.Person;
public class Employee extends Person {
  double salary;
  public String getDetail() {
    String s = super.getDetail() + "," + salary;
    return s;
  }
}
```



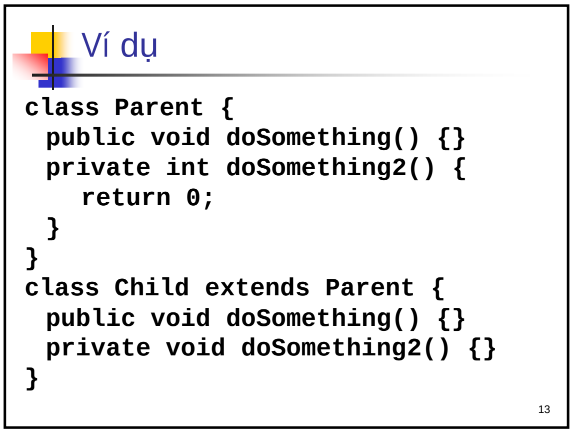
1. Định nghĩa lại hay ghi đè (3)

- Một số quy định



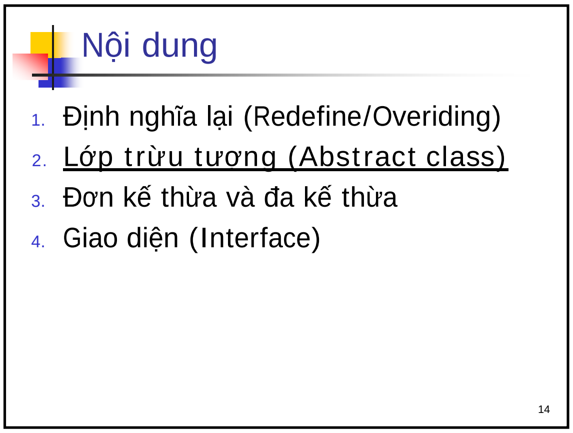
Ví dụ

```
class Parent {
  public void doSomething() {}
  protected int doSomething2() {
     return 0;
  }
}
class Child extends Parent {
  protected void doSomething() {}
  protected void doSomething2() {}
}
```



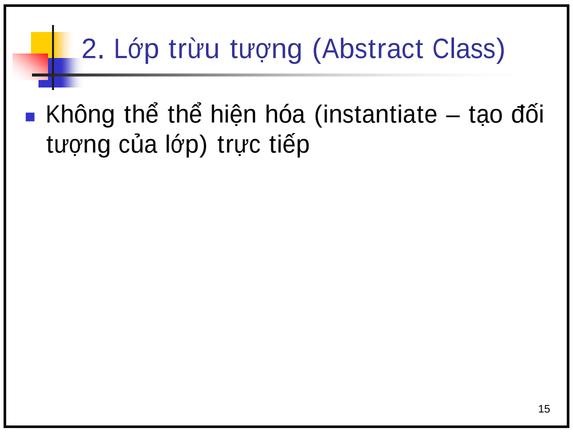
Ví dụ

```
class Parent {
  public void doSomething() {}
  private int doSomething2() {
     return 0;
  }
}
class Child extends Parent {
  public void doSomething() {}
  private void doSomething2() {}
}
```



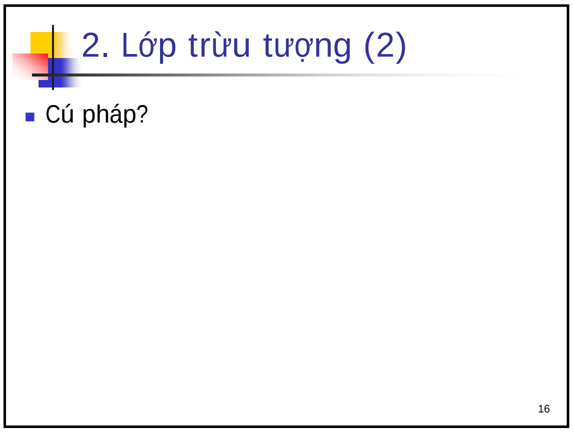
Nội dung

1. Định nghĩa lại (Redefine/Overiding)
2. Lớp trừu tượng (Abstract class)
3. Đơn kế thừa và đa kế thừa
4. Giao diện (Interface)



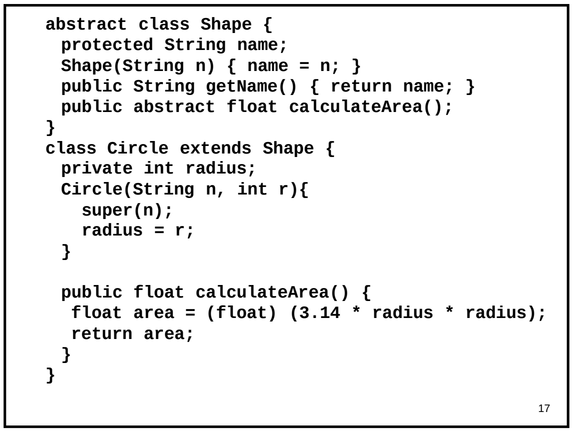
2. Lớp trừu tượng (Abstract Class)

- Không thể thể hiện hóa (instantiate – tạo đối tượng của lớp) trực tiếp



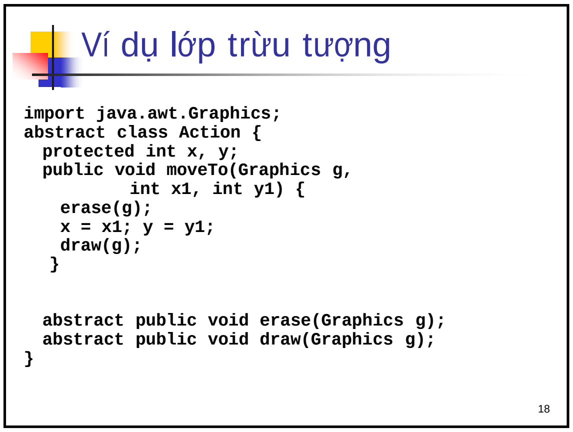
2. Lớp trừu tượng (2)

- Cú pháp?



```
abstract class Shape {
  protected String name;
  Shape(String n) { name = n; }
  public String getName() { return name; }
  public abstract float calculateArea();
}
class Circle extends Shape {
  private int radius;
  Circle(String n, int r){
    super(n);
    radius = r;
  }

  public float calculateArea() {
   float area = (float) (3.14 * radius * radius);
   return area;
  }
}
```



Ví dụ lớp trừu tượng

```
import java.awt.Graphics;
abstract class Action {
  protected int x, y;
  public void moveTo(Graphics g,
           int x1, int y1) {
    erase(g);
    x = x1; y = y1;
    draw(g);
   }


  abstract public void erase(Graphics g);
  abstract public void draw(Graphics g);
}
```

Ví dụ lớp trừu tượng (2)

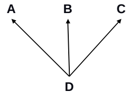
```
class Circle extends Action {
   int radius;
   public Circle(int x, int y, int r) {
     super(x, y); radius = r;
   }
   public void draw(Graphics g) {
     System out println("Draw circle at ("
                             + x + "," + y + ")");
     g.drawOval(x-radius, y-radius,
                  2*radius, 2*radius);
   }
   public void erase(Graphics g) {
      System.out.println("Erase circle at ("
                              + x + "," + y + ")");
   }
}
```

Nội dung

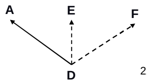
1. Định nghĩa lại (Redefine/Overiding)
2. Lớp trừu tượng (Abstract class)
3. Đơn kế thừa và đa kế thừa
4. Giao diện (Interface)

Đa kế thừa và đơn kế thừa

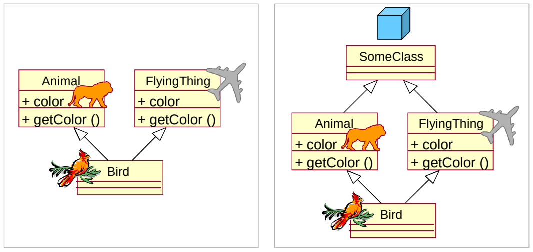
- Đa kế thừa (Multiple Inheritance)
  - khác



- Đơn kế thừa (Single Inheritance)

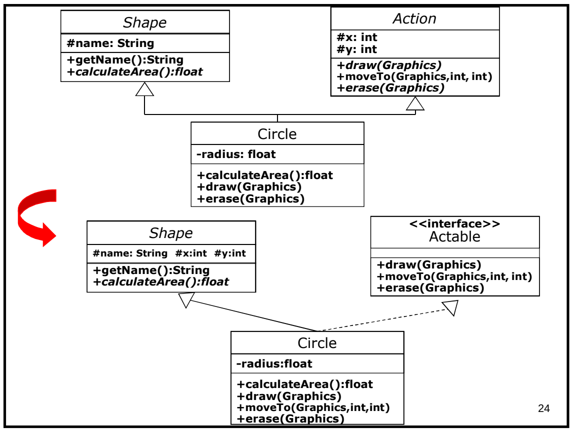


Vấn đề gặp phải trong Đa kế thừa



Nội dung

1. Định nghĩa lại (Redefine/Overiding)
2. Lớp trừu tượng (Abstract class)
3. Đơn kế thừa và đa kế thừa
4. Giao diện (Interface)



Shape
#name: String
+getName():String
+calculateArea():float

Action
#x: int
#y: int
+draw(Graphics)
+moveTo(Graphics,int, int)
+erase(Graphics)

Circle
-radius: float
+calculateArea():float
+draw(Graphics)
+erase(Graphics)

Shape
#name: String #x:int #y:int
+getName():String
+calculateArea():float

<<interface>>
Actable
+draw(Graphics)
+moveTo(Graphics,int, int)
+erase(Graphics)

Circle
-radius:float
+calculateArea():float
+draw(Graphics)
+moveTo(Graphics,int,int)
+erase(Graphics)

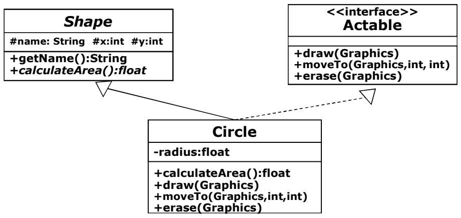
4. Giao diện

Không thể thể hiện hóa (instantiate) trực tiếp

4. Giao diện (2)

- Cú pháp?

Ví dụ

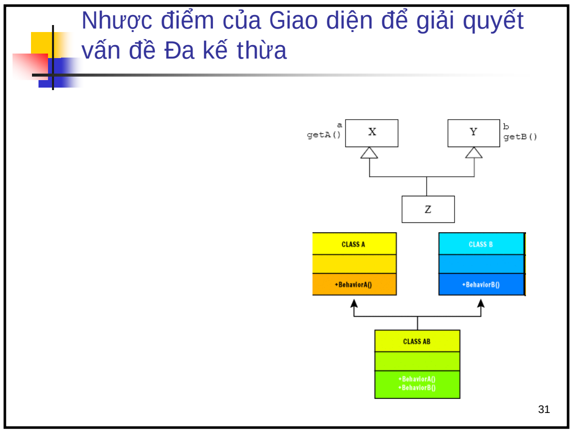


```
import java.awt.Graphics;
abstract class Shape {
  protected String name;
  protected int x, y;
  Shape(String n, int x, int y) {
      name = n; this.x = x; this.y = y;
  }
  public String getName() {
      return name;
  }
  public abstract float calculateArea();
}
interface Actable {
  public void draw(Graphics g);
  public void moveTo(Graphics g, int x1, int y1);
  public void erase(Graphics g);
}
```

```
class Circle extends Shape implements Actable {
   private int radius;
  public Circle(String n, int x, int y, int r){
       super(n, x, y); radius = r;
  }
   public float calculateArea() {
       float area = (float) (3.14 * radius * radius);
       return area;
   }
   public void draw(Graphics g) {
      System out println("Draw circle at ("
                              + x + “," + y + ")");
    g.drawOval(x-radius,y-radius,2*radius,2*radius);
   }
   public void moveTo(Graphics g, int x1, int y1){
       erase(g); x = x1; y = y1; draw(g);
   }
   public void erase(Graphics g) {
        System out println(“Erase circle at ("
                              + x + “," + y + ")");
     // paint the region with background color...
   }
}
```

Lớp trừu trượng vs. Giao diện

Lớp trừu trượng	Giao diện



Nhược điểm của Giao diện để giải quyết vấn đề Đa kế thừa

31