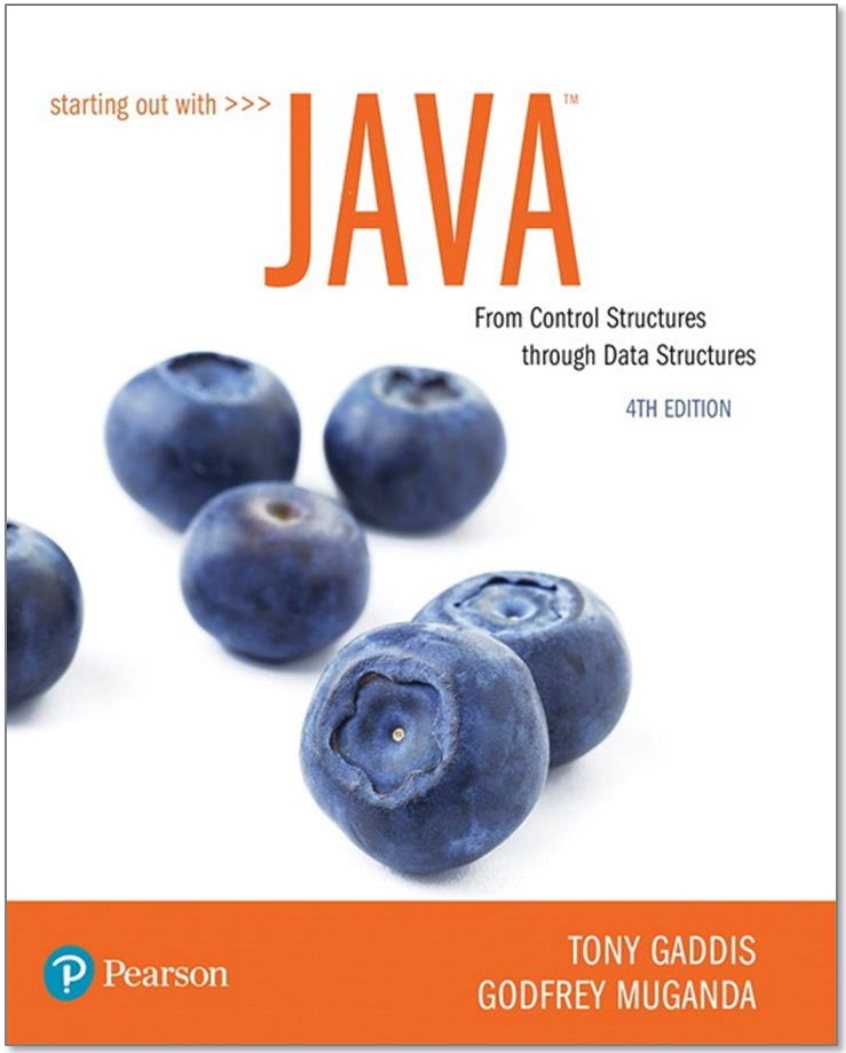
STARTING OUT WITH JAVA™

4th Edition



Chương 10

Kế thừa

Pearson

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Nội dung

- Thừa kế là gi?
- Gọi hàm khởi tạo lớp cha
- Ghi đè phương thức lớp cha
- Các thành phần `protected`
- Chuỗi kế thừa
- Lớp `Object`
- Tính đa hình
- Các lớp và phương thức trừu tượng
- Giao diện (interface)
- Các lớp ẩn danh (Anonymous)

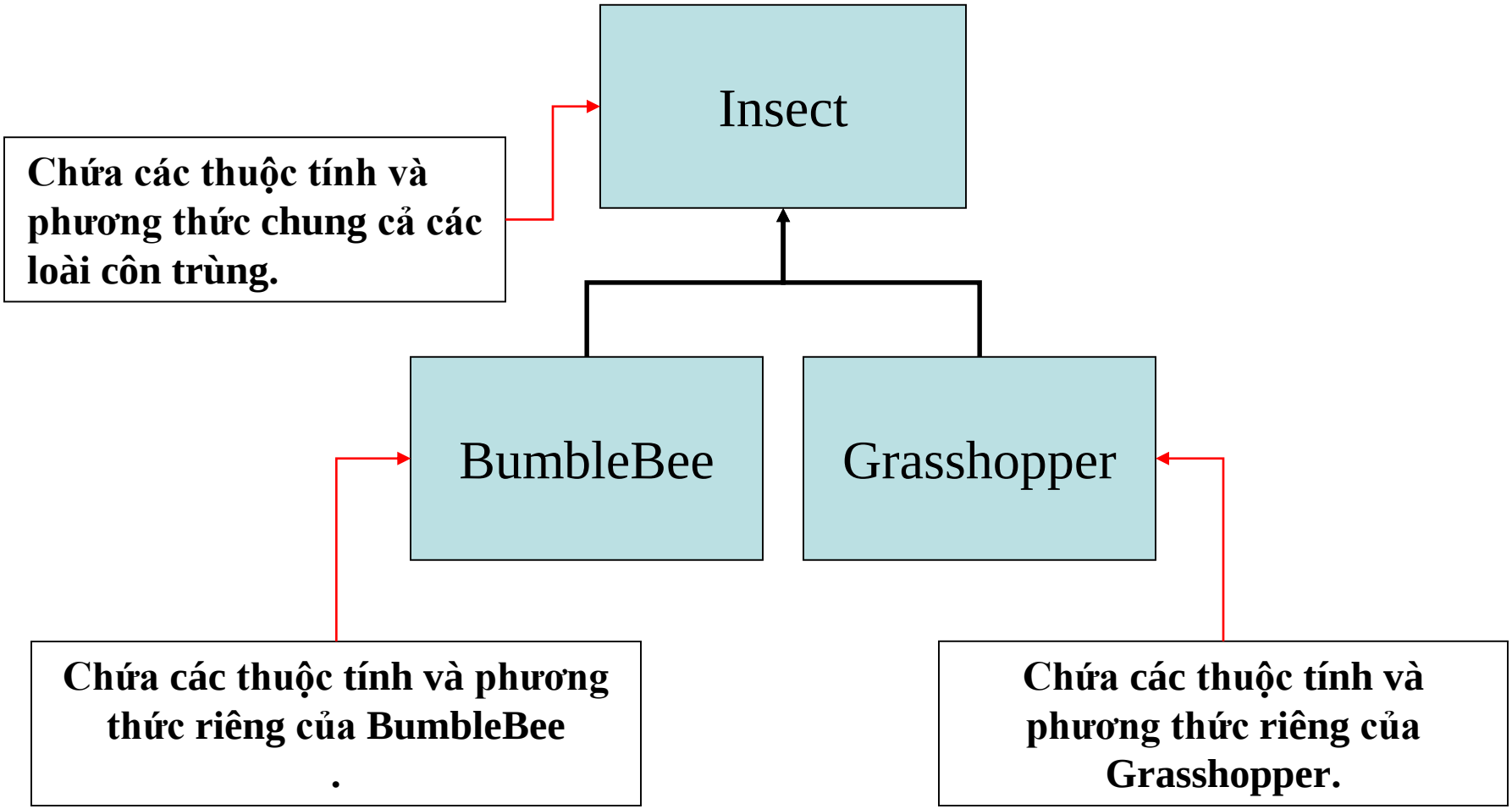
Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Kế thừa là gì?
Tổng quát hóa vs Chuyên môn hóa

- Các đối tượng trong thế giới thực thường là phiên bản chuyên biệt của các đối tượng khác tổng quát hơn.
- Thuật ngữ “insect“ (côn trùng) mô tả một loại sinh vật rất chung chung với nhiều đặc điểm.
- Châu chấu và con ong là loại côn trùng
  - Chúng có những đặc điểm chung của loài côn trùng.
  - Tuy nhiên, chúng có những đặc điểm riêng.
    - Châu Chấu có khả năng nhảy
    - Con Ong thì có ngòi (kim).
- Châu Chấu và con Ong là sự chuyên biệt hóa của loài côn trùng.

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Kế thừa (Inheritance)



Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Mối quan hệ là một <is a> (1)

- Mối quan hệ giữa lớp cha và lớp kế thừa được gọi là mối quan hệ **là** “is a".
  - Châu chấu **là** một côn trùng.
  - Xe hơi **là** một phương tiện.
- Trong lập trình hướng đối tượng, kế thừa được sử dụng để tạo mối quan hệ “**is a**" giữa các lớp

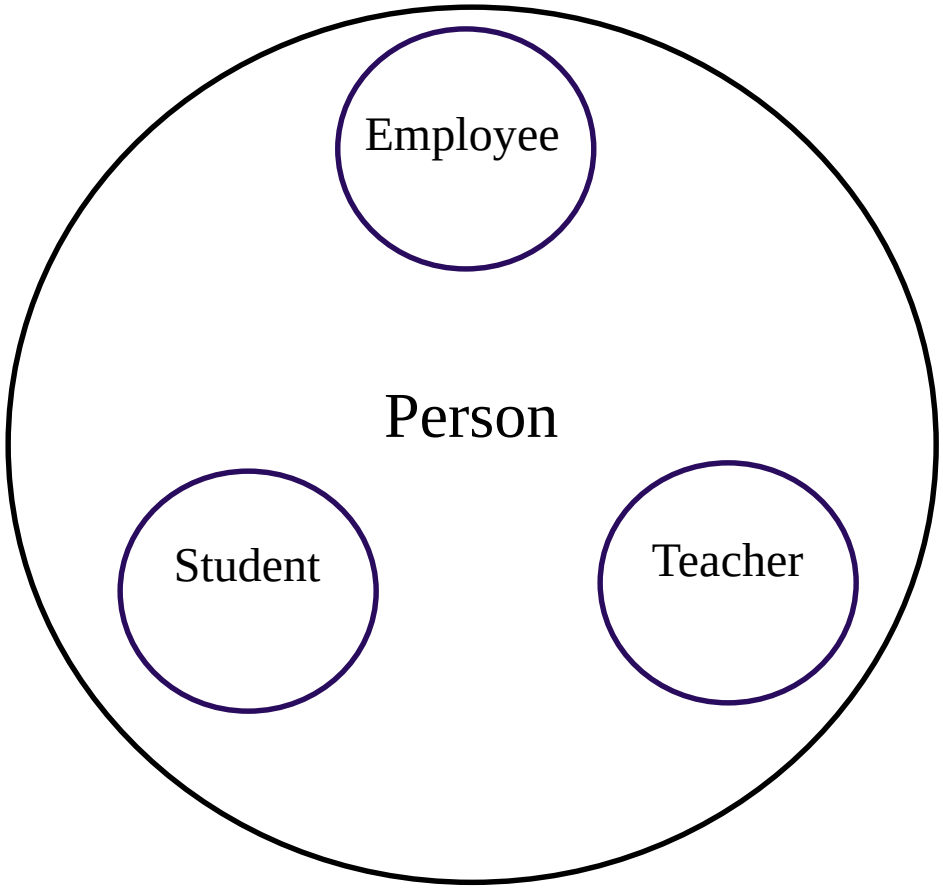
Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Mối quan hệ là một <is a> (1)

Mối quan hệ là một <is a> (2)

- Lớp cha và lớp con có tính tương đồng
  - Lớp sinh viên kế thừa từ lớp **Nguoi**
  - Một sinh viên là một người

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved



Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

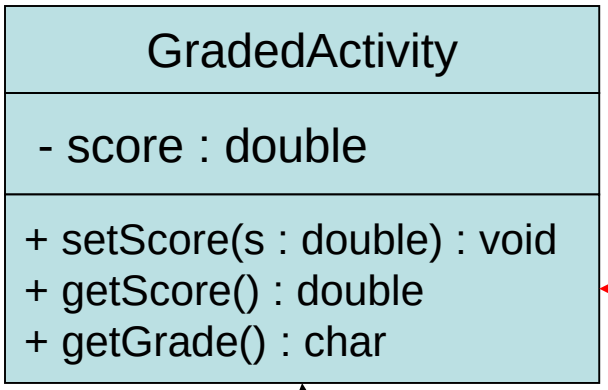
Kế thừa (Inheritance)

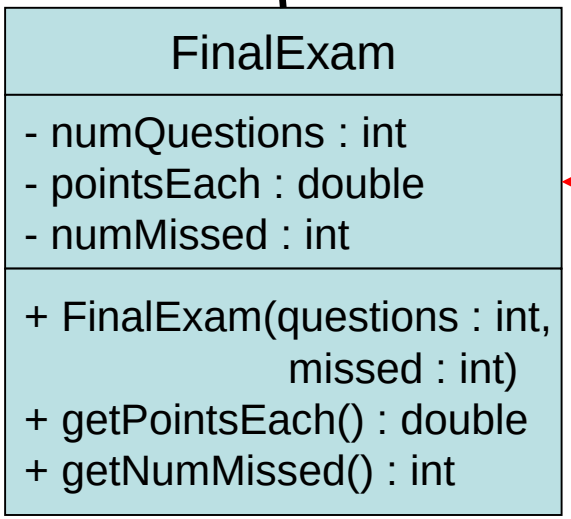
- Lớp con kế thừa các thuộc tính và phương thức của lớp cha → không cần viết lại.
  - Các sinh viên đều phải ăn
  - Các giáo viên đều phải ăn
  - Con người phải ăn
- Các thuộc tính và phương thức mới có thể thêm vào lớp con.
  - Sinh viên phải đi học, giáo viên phải đi dạy.
  - Con người đi học và đi dạy?
- Từ khóa *extends* được dùng để khai báo lớp con.

```
public class FinalExam extends GradedActivity
```

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Ví dụ `GradedActivity`





Chứa các thuộc tính và phương thức dùng chung.

Chứa các thuộc tính và phương thức dành riêng cho lớp `FinalExam`.
Kế thừa tất cả các thuộc tính và phương thức được khai báo là protected và public từ lớp GradeActivity

- Example:
  - GradedActivity.java,
  - GradeDemo.java,
  - FinalExam.java,
  - FinalExamDemo.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

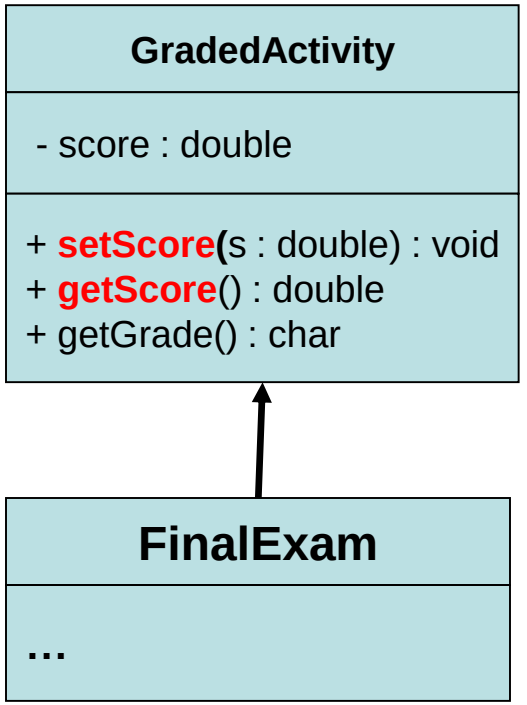
Kế thừa, thuộc tính và phương thức (1)

- Lớp cha có các thành phần khai báo **private**
  - không cho phép thừa kế.
  - tồn tại trong bộ nhớ khi đối tượng của lớp con được tạo.
  - lớp con chỉ có thể truy cập bằng các phương thức được khai báo public của lớp cha
- Lớp cha có các thành phần khai báo **public**
  - Cho phép lớp con kế thừa

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Kế thừa, thuộc tính và phương thức (2)

- Khi một instance của lớp con được tạo, các phương thức non-private của lớp cha sẽ có thể được gọi thông qua đối tượng của lớp con.




```
FinalExam exam = new FinalExam();

exam.setScore(85.0);

System.out.println("Score = " + exam.getScore());
```

Pearson

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Kế thừa và Phương thức khởi tạo.

- Phương thức khởi tạo không được kế thừa.
- Khi một lớp con được khởi tạo, phương thức khởi tạo mặc định của lớp cha được thực thi đầu tiên.
- Ví dụ:
  - SuperClass1.java
  - SubClass1.java
  - ConstructorDemo1.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Hàm khởi tạo của lớp cha

- Từ khóa `super` đại diện cho lớp cha của một đối tượng.
- Phương thức khởi tạo lớp cha có thể được gọi một cách tường minh (explicit) từ lớp con bằng cách sử dụng từ khóa `super`.
- Ví dụ:
  - SuperClass2.java, SubClass2.java, ConstructorDemo2.java
  - Rectangle.java, Cube.java, CubeDemo.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Gọi phương thức khởi tạo của lớp cha

- Nếu phương thức khởi tạo có tham số được khai báo trong lớp cha,
  - Lớp cha phải cung cấp phương thức khởi tạo không tham số, hoặc
    - lớp con phải cung cấp mật phương thức khởi tạo, và
    - lớp con phải gọi hàm khởi tạo của lớp cha.
- *Lệnh gọi phương thức khởi tạo của lớp cha phải được khai báo tại dòng lệnh đầu tiên trong phương thức khởi tạo của lớp con.*

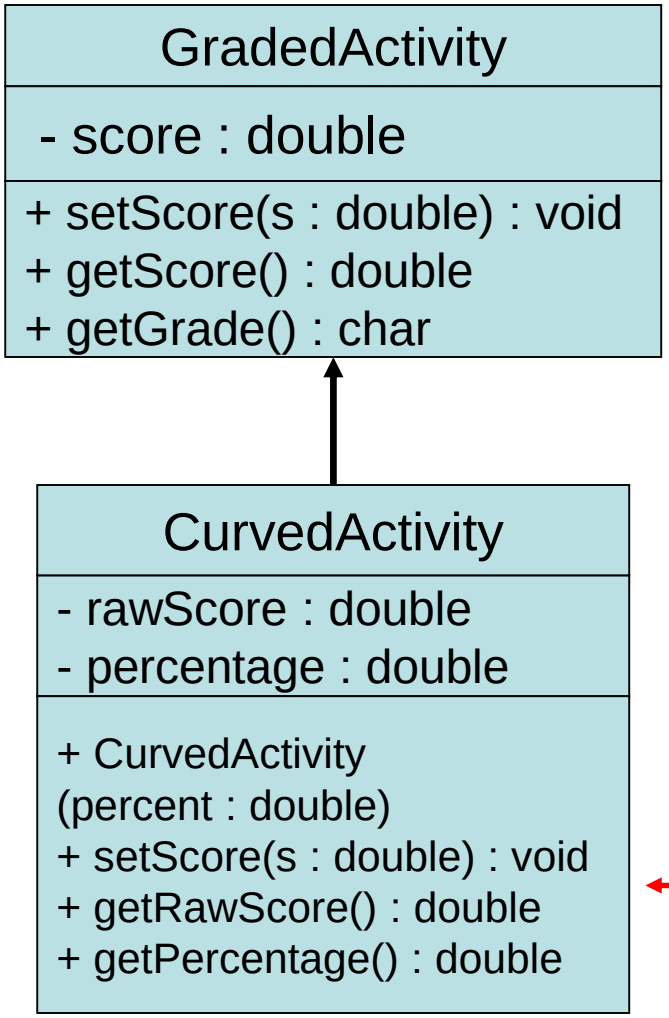
Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Ghi đè các phương thức của lớp cha (1)

- Lớp con có thể có phương thức cùng tên cùng danh sách tham số với phương thức của lớp cha.
  - Hiện tượng này gọi là ghi đè (*overriding)*.
- Ví dụ:
  - GradedActivity.java, CurvedActivity.java, CurvedActivityDemo.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Ghi đè các phương thức của lớp cha (2)



Phương thức này là một phiên bản riêng của phương thức setScore trong lớp cha, `GradedActivity`.

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Ghi đè các phương thức của lớp cha (3)

- Một phương thức của lớp con ghi đè một phương thức của lớp cha phải thỏa mãn các điều kiện sau đây:
  - Giống tên và danh sách tham số.
  - Phải có chú thích là @Override trên phương thức.

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Ghi đè các phương thức của lớp cha (4)

- Một phương thức của lớp con có thể gọi phương thức ghi đè của lớp cha thông qua từ khóa `supper`

```
super.setScore(rawScore * percentage);
```

- Phân biệt overloading và overriding:

  - **Overloading** : Cho phép phương thức trong cùng một lớp có thể được cùng tên nhưng chúng bắt buộc phải thỏa điều kiện là khác danh sách tham số.
    - Danh sách tham số khác nhau khi kích thước danh sách khác nhau hoặc kiểu dữ liệu của phần tử cùng vị trí khác nhau.

  - **Overriding**: khi một phương thức ghi đè một phương thức khác, cả hai phải cùng tên và danh sách tham số.

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Ghi đè các phương thức của lớp cha (5)

- Ví dụ:
  - SuperClass3.java,
  - SubClass3.java,
  - ShowValueDemo.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Quản lý ghi đè

- Một phương thức sử dụng từ khóa `final` sẽ chặn không cho lớp con ghi đè.

```
public final void message()
```

- Nếu một lớp con cố gắng ghi đè một phương thức `final`, trình biên dịch sẽ tạo ra lỗi.

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Các thành phần protected (1)

- Các thành phần `protected` của lớp:
  - có thể được truy cập bằng các phương thức của lớp con, và
  - bằng các phương thức trong cùng package với lớp.
- Ví dụ:
  - GradedActivity2.java
  - FinalExam2.java
  - ProtectedDemo.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Các thành phần protected (2)

- Sử dụng `protected` thay vì `private` thực hiện một số tác vụ dễ dàng hơn.
- Tuy nhiên, bất kỳ một một lớp nào nằm cùng một package, đều có quyền truy cập không hạn chế vào các thành phần `protected`.
- Bất kỳ phương thức trong cùng package đều có thể truy cập .

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Access Specifiers

Access Modifier	Accessible to a subclass inside the same package?	Accessible to all other classes inside the same package?
default (no modifier)	Yes	Yes
Public	Yes	Yes
Protected	Yes	Yes
Private	No	No

Access Modifier	Accessible to a subclass outside the package?	Accessible to all other classes outside the package?
default (no modifier)	No	No
Public	Yes	Yes
Protected	Yes	No
Private	No	No

Pearson

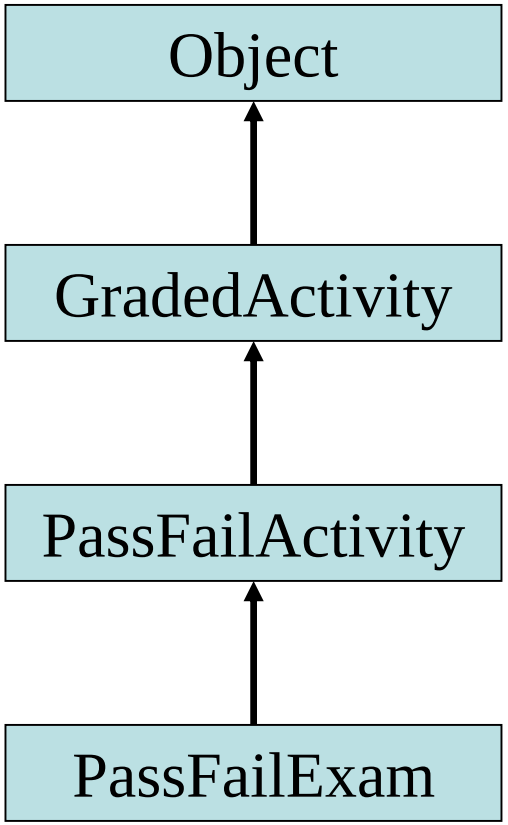
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Chuỗi kế thừa (1)

- Một lớp con vẫn có thể cho phép các lớp khác kế thừa, điều này tạo ra chuỗi kế thừa.

Ví dụ:

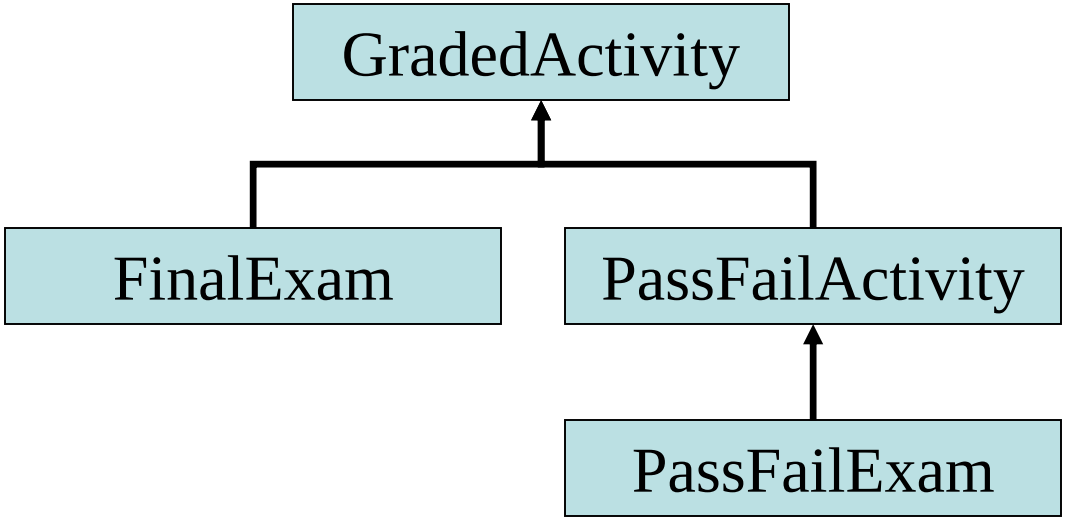
GradedActivity.java
PassFailActivity.java
PassFailExam.java
PassFailExamDemo.java



Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Chuỗi kế thừa (2)

- Cấu trúc phân cấp hình cây, biểu diễn mối quan hệ kế thừa giữa các lớp.



Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Lớp `Object` (1)

- Lớp `Object` là lớp gốc trên cùng của tất cả các cây phân cấp kế thừa.
- Được định nghĩa trong package `java.lang`.
- Nếu một lớp không được định nghĩa là lớp con của một lớp khác thì mặc định là lớp con trực tiếp của lớp `Object`

```
public class MyClass
      {
         // This class is derived from Object.
      }
```

Pearson

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Lớp `Object` (2)

- Chứa một số phương thức hữa ích kế thừa lại cho tất cả các lớp
  - Ví dụ: `toString()`, `equals()`,...
- Ví dụ: ObjectMethods.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Đa hình (1)

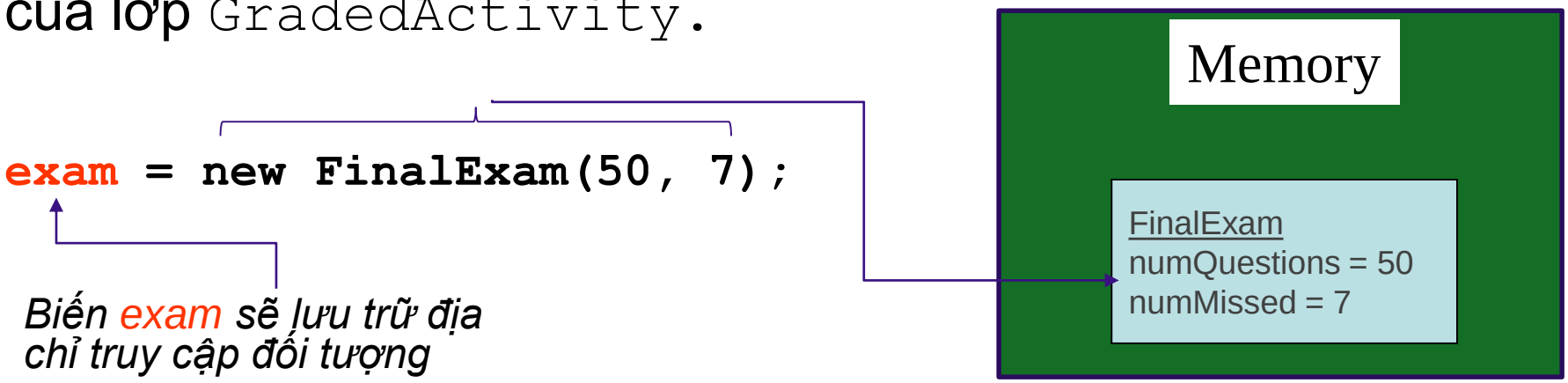
Một biến tham chiếu đến đối tượng của lớp cha thì cũng có thể tham chiếu đến các đối tượng có nguồn gốc từ lớp cha.

Ví dụ:

```
GradedActivity exam; // khai báo biến exam tham chiếu đến đối tượng lớp GradedActivity.

exam = new GradedActivity(); // khởi tạo đối tượng GradedActivity và gán tham chiếu cho biến exam.
```

Một đối tượng của lớp `FinalExam` cũng là một đối tượng của lớp `GradedActivity`.



Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Đa hình (3)

- Các tham chiếu khác minh họa về tính đa hình:

```
GradedActivity exam1 = new FinalExam(50, 7);
GradedActivity exam2 = new PassFailActivity(70);
GradedActivity exam3 = new PassFailExam(100, 10, 70);
```

- Lớp `GradedActivity` có 03 phương thức: `setScore`, `getScore`, và `getGrade`.
- Một biến được khai báo là tham chiếu đến đối tượng `GradedActivity` thì chỉ có thể gọi được 03 phương thức của lớp `GradedActivity`.

```
GradedActivity exam = new PassFailExam(100, 10, 70);
System.out.println(exam.getScore()); // This works.
System.out.println(exam.getGrade()); // This works.
System.out.println(exam.getPointsEach()); // ERROR!
```

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Đa hình (3)

Biến exam khai báo sẽ tham chiếu đến đối tượng GradedActivity nhưng lại tham chiếu đến đối tượng PassFailExam:

```
GradedActivity exam = new PassFailExam(100, 10, 70);
```

- Tính chất đa hình. Tuy nhiên, biến exam chỉ có thể gọi đến 03 phương thức của lớp `GradedActivity` và lỗi khi gọi phương thức của lớp `PassFailExam`.

```
System.out.println(exam.getScore()); // This works.
System.out.println(exam.getGrade()); // This works.
System.out.println(exam.getPointsEach()); // ERROR!
```

- Điều này xảy ra là vì ngay từ đầu **exam** đã được khai báo là sẽ tham chiếu đến đối tượng `GradedActivity`. Để khắc phục lỗi, chúng ta cần chuyển đổi kiểu tham chiếu (cast) cho biến **exam**.

```
System.out.println(exam.getPointsEach()); // ERROR!

double pointsEach = ((PassFailExam)exam).getPointsEach();
System.out.println(pointsEach);
```

Pearson

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Polymorphism and Dynamic Binding

- Nếu đối tượng của lớp con đã ghi đè lên một phương thức của lớp cha:
  - Nếu biến thực hiện gọi lệnh đến phương thức đó, phiên bản phương thức của lớp con sẽ thực thi.

```
GradedActivity exam = new PassFailActivity(60);
exam.setScore(70);
System.out.println(exam.getGrade());
```

- Java thực hiện gắn kết động khi một biết tham chiếu đa hình
- Khi chạy chương trình máy ảo Java xác định phương thức nào sẽ gọi, tùy thuộc vào đối tượng mà biến tham chiếu.

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Đa hình (4)

- Đây là kiểu đối tượng, không phải kiểu tham chiếu, sẽ xác định phương thức nào được gọi.
- Ví dụ:
  - Polymorphic.java
- Bạn không thể gán một đối tượng lớp cha cho một biến tham chiếu lớp con.

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Lớp trừu tượng (Abstract Classes)

- Lớp trừu tượng không cho phép tạo đối tượng, nhưng vẫn có thể được thừa kế từ lớp khác.
- Thông thường, một lớp trừu tượng được sử dụng như là lớp cha cho các lớp khác.
- Lớp trừu tượng được dùng để định nghĩa các "khái niệm chung chung" đóng vai trò làm lớp cơ sở cho các lớp "cụ thể" khác.
- Cú pháp: khai báo với từ khóa **`abstract`**.

```
public abstract class ClassName
```

Pearson

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Phương thức trừu tượng(1)

- Phương thức trừu tượng không triển khai xử lý tính toán (phương thức không có phần body hay phương thức không thực thi các câu lệnh) và phải được ghi đè ở lớp con.

```
public abstract float calculateArea();
```

- Các phương thức được ghi đè bởi lớp con sẽ triển khai các xử lý tính toán.
- Nếu một lớp con không ghi đè một phương thức trừu tượng, thì sẽ xảy ra lỗi trình biên dịch.
- Ví dụ:
  - Student.java, CompSciStudent.java, CompSciStudentDemo.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Phương thức trừu tượng(2)

- Lớp trừu tượng phải có ít nhất một phương thức trừu tượng (abstract method).
- Lớp không phải trừu tượng thì không thể tồn tại các phương thức trừu tượng.

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Giao diện - Interfaces (1 of 3)

- Một interface tương tự như một lớp trừu tượng, chứa các phương thức trừu tượng.
  - Nó không thể tạo ra đối tượng, và
  - Các phương thức trong interface phải viết ở các lớp khác.
- Mục đích của giao diện là chỉ định hành vi cho các lớp khác.
- Một interface giống như một lớp, ngoại trừ:
  - từ khóa interface được sử dụng thay cho từ khóa class, và
  - phương thức được định nghĩa trong `interface` không có phần thân, chỉ có phần header được kết thúc bằng dấu chấm phẩy.

Pearson

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Giao diện - Interfaces (2 of 3)

- Dạng chung của interace như sau:

```
public interface InterfaceName
{
   (Method headers...)
}
```

- Tất cả các phương thức trong các interface đều phải là phương thức `public`.
- Một lớp có thể thực thi một hoặc nhiều interface.

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Giao diện - Interfaces (3 of 3)

- Cú pháp thực thi giao diện:

```
public class FinalExam3 extends
   GradedActivity implements Relatable
```

- Ví dụ:
  - GradedActivity.java
  - Relatable.java
  - FinalExam3.java
  - InterfaceDemo.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Các thuộc tính trong interface

- Một interface có thể khai báo các thuộc tính:
  - Các thuộc tính này được xem như là các `final` và `static`.
- Bởi vì đây là các `final` → cần phải khởi tạo các giá trị cho thuộc tính.

```
public interface Doable
{
  int FIELD1 = 1, FIELD2 = 2;
  (Method headers...)
}
```

- Thuộc tính `FIELD1` và `FIELD2` là các biến `final static int`
- Lớp thực thi interface này đều có quyền truy cập vào các biến này.

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

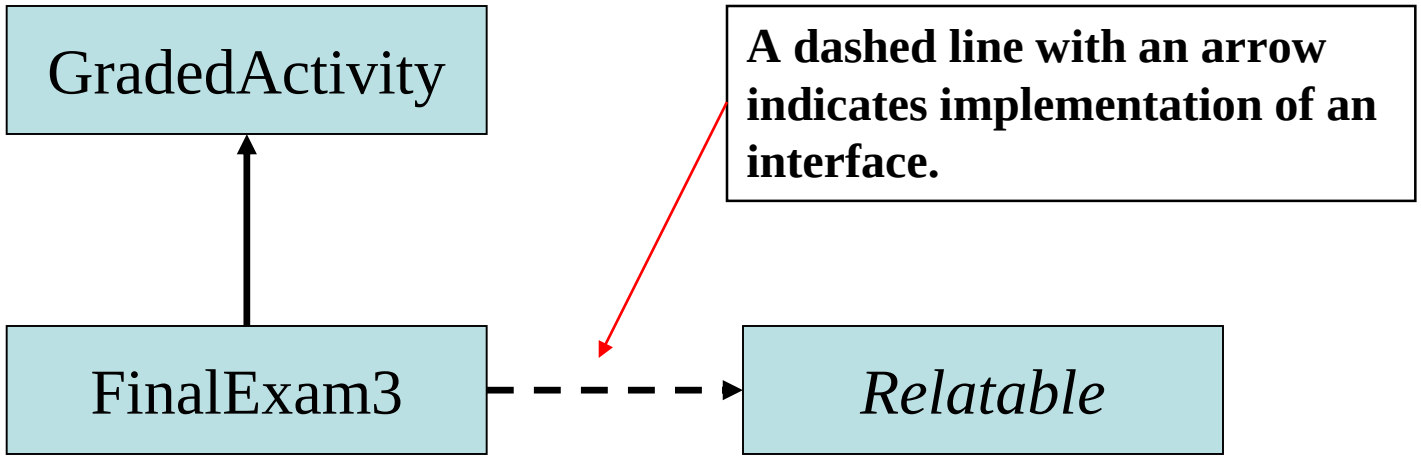
Thực thi nhiều Interface

- Một lớp chỉ được kế thừa một lớp khác.
- Một lớp có thể thực thi nhiều interface.
- Khi một lớp thực thi interface, nó phải thực thi tất cả các phương thức được chỉ ra bởi các interface đó
- Khi thực thi nhiều interface thì các interface sẽ được ngăn cách nhau bởi dấu phẩy.
- Cú pháp:

```
public class MyClass implements Interface1,
                                 Interface2,
                                 Interface3
```

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Mô hình Interface trong sơ đồ UML



Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Polymorphism with Interfaces (1 of 3)

- Java cho phép khai báo một biến tham chiếu có kiểu là interface.
- Một biến tham chiếu interface có thể tham chiếu đến bất kỳ đối tượng nào triển khai interface đó, bất kể loại lớp của nó là gì.
- Ví dụ về đa hình:
  - RetailItem.java
  - CompactDisc.java
  - DvdMovie.java
  - PolymorphicInterfaceDemo.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Polymorphism with Interfaces (2 of 3)

- Trong ví dụ , khai báo 02 biến tham chiếu `RetailItem`.
- Biến `item1` tham chiếu đến một đối tượng `CompactDisc` và biến`item2` tham chiếu đến một đối tượng `DvdMovie`.
- Khi một lớp thực hiện một interface, một mối quan hệ kế thừa được gọi là kế thừa interface được thiết lập.
  - a `CompactDisc` object ***is a*** `RetailItem`, and
  - a `DvdMovie` object ***is a*** `RetailItem`.

Pearson

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Polymorphism with Interfaces (3 of 3)

- Tham chiếu đến một interface có thể trỏ đến bất kỳ lớp nào triển khai interface đó..
- Không thể tạo một đối tượng của interface.

```
RetailItem item = new RetailItem(); // ERROR!
```

Pearson

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Phương thức mặc định

- Bắt đầu từ Java 8, các interface có thể có các phương thức mặc định.

```
public interface Displayable{
   default void display(){
      System.out.println("This is the default display method.");
   }
}
```

- Ví dụ:
  - Displayable.java
  - Person.java
  - InterfaceDemoDefaultMethod.java

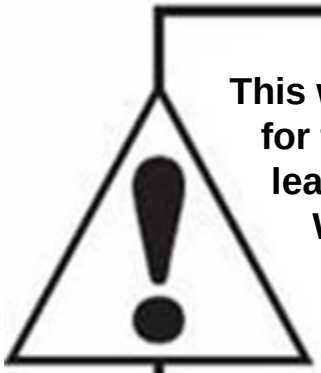
Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Các lớp ẩn danh

- Lớp lồng nhau (inner class) là một lớp được khai báo trong một lớp khác.
- Một lớp ẩn danh (anonymous inner class) là một lớp không có tên gọi.
- Một lớp ẩn danh được tạo ra thông qua thực thi `interface`, hoặc kế thừa từ lớp khác.
- Lợi ích khi sử dụng lớp ẩn danh: mã code dễ đọc và dễ bảo trì
- Ví dụ:
  - IntCalculator.java
  - AnonymousClassDemo.java

Pearson
Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved

Copyright



This work is protected by United States copyright laws and is provided solely for the use of instructions in teaching their courses and assessing student learning. dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.

Copyright © 2019, 2016, 2012 Pearson Education, Inc. All Rights Reserved