CHAPTER 7 Inheritance

Shin-Jie Lee (李信杰)
Associate Professor
Computer and Network Center
Department of Computer Science and Information Engineering
National Cheng Kung University

成功大學
National Cheng Kung University

Inheritance

- The sharing of attributes and operations among classes based on a hierarchical relationship
  - It allows code to be *reused*, without having to copy it into the definitions of the derived classes

- **Is-a** relationship

Introduction to Inheritance

- The original class is called the *base class*
- The new class is called a *derived class*

Derived Class (Subclass)

- ❑ Members of a class that are declared **private** are not inherited by subclasses of that class.

- ❑ Only members of a class that are declared **protected** or **public** are inherited by subclasses declared in a package other than the one in which the class is declared.

Lab

```
import java.util.Date;

public class Employee {

  protected String name;
  protected Date hireDate;

  public Employee(){}

  public Employee(String theName, Date theDate){
        name = theName;
        hireDate = theDate;
  }
  public Date getHireDate(){
        return hireDate;
  }

  public String getName(){
        return name;
  }

}
```

Lab

```
import java.util.Date;

public class HourlyEmployee extends Employee{
 private double wageRate;

 public HourlyEmployee(String theName, Date theDate, double rate){
        name = theName;
        hireDate = theDate;
        wageRate = rate;
 }

}
```

Lab

```
import java.util.Date;

public class Company {

  public static void main(String[] args){

    HourlyEmployee hourlyEmployee = new HourlyEmployee("Josephine",
         new Date(114,0,1), 100);

    System.out.println(hourlyEmployee.getName());

  }
}
```

Overriding a Method Definition

- Although a derived class inherits methods from the base class, it can **change** or ***override*** an inherited method if necessary

Lab

```
import java.util.Date;

public class HourlyEmployee extends Employee{
 private double wageRate;

 public HourlyEmployee(String theName, Date theDate, double rate){
        name = theName;
        hireDate = theDate;
        wageRate = rate;
 }
 public String getName(){
        return "Hourly Employee:" + name;
 }
}
```

Then run Company again!

The `super` Constructor

- A derived class uses a constructor from the base class to initialize all the data inherited from the base class
  - In order to invoke a constructor from the base class, it uses a special syntax:

```
public derivedClass(int p1, int p2, double p3)
{
  super(p1, p2);
  instanceVariable = p3;
}
```

  - In the above example, `super(p1, p2);` is a call to the base class constructor

Lab

```
import java.util.Date;

public class HourlyEmployee extends Employee{
 private double wageRate;

 public HourlyEmployee(String theName, Date theDate, double rate){

          super(theName,theDate);   Step 1: revise code here

          wageRate = rate;
 }
 public String getName(){
        return "Hourly Employee:" + name;
 }
}
```

Step 2: then run the Company again!

The `this` Constructor

- Often, a no-argument constructor uses **`this`** to invoke an explicit-value constructor
  - No-argument constructor (invokes explicit-value constructor using **`this`** and default arguments):

```
public ClassName()
{
   this(argument1, argument2);
}
```

  - Explicit-value constructor (receives default values):

```
public ClassName(type1 param1, type2 param2)
{
   . . .
}
```

Tip: An Object of a Derived Class Has More than One Type

- **An object of a derived class has the type of every one of its ancestor classes**
  - Therefore, **an object of a derived class can be assigned to a variable of any ancestor type**

Lab

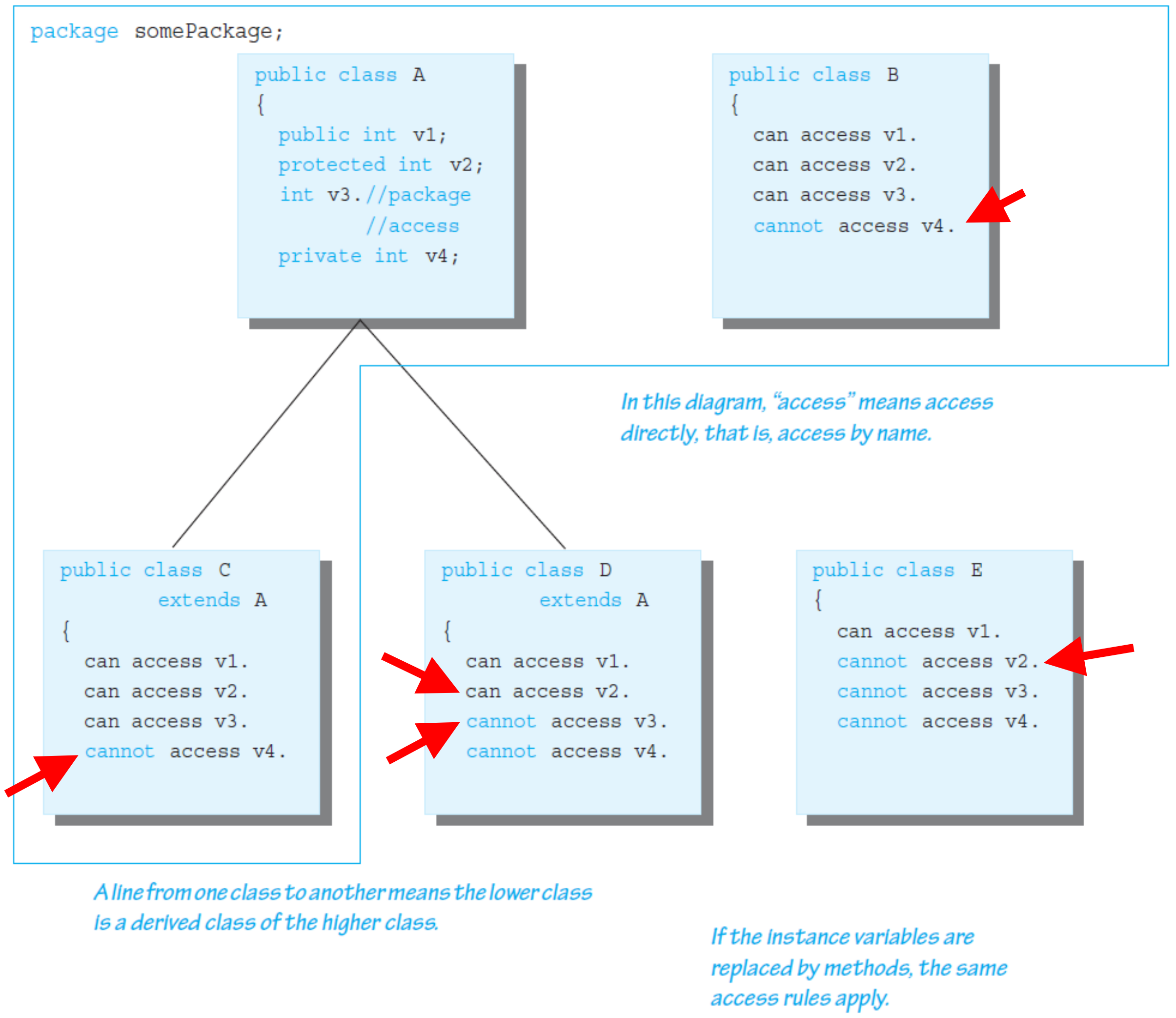
```
import java.util.Date;

public class Company {

  public static void main(String[] args){

    HourlyEmployee hourlyEmployee = new HourlyEmployee("Josephine",
       new   Date(114,0,1), 100);

    System.out.println(hourlyEmployee.getName());

    Employee someEmploy = hourlyEmployee;
    printHireDate(someEmploy);
  }

  public static void printHireDate(Employee someEmploy){
    System.out.println(someEmploy.getHireDate());
  }
}
```

Modifiers

Modifier	Class	Package	Subclass	World
public	V	V	V	V
protected	V	V	V	X
default (package access)	V	V	X	X
private	V	X	X	X

Access Modifiers

Display 7.9 Access Modifiers



```
package somePackage;

public class A
{
  public int v1;
  protected int v2;
  int v3.//package
         //access
  private int v4;
```

```
public class B
{
  can access v1.
  can access v2.
  can access v3.
  cannot access v4.
```

In this diagram, "access" means access directly, that is, access by name.

```
public class C
        extends A
{
  can access v1.
  can access v2.
  can access v3.
  cannot access v4.
```

```
public class D
        extends A
{
  can access v1.
  can access v2.
  cannot access v3.
  cannot access v4.
```

```
public class E
{
  can access v1.
  cannot access v2.
  cannot access v3.
  cannot access v4.
```

A line from one class to another means the lower class is a derived class of the higher class.

If the instance variables are replaced by methods, the same access rules apply.

Reference

- “Absolute Java”. Walter Savitch and Kenrick Mock. Addison-Wesley; 5 edition. 2012
- “Java How to Program”. Paul Deitel and Harvey Deitel. Prentice Hall; 9 edition. 2011.
- “A Programmers Guide To Java SCJP Certification: A Comprehensive Primer 3rd Edition”. Khalid Mughal, Rolf Rasmussen. Addison-Wesley Professional. 2008