Java 应用与开发
面向对象编程进阶 PART1

王晓东
wangxiaodong@ouc.edu.cn

中国海洋大学

September 23, 2018

学习目标

1. 掌握 Java 包、继承、访问控制、方法重写的概念、机制和使用方法
2. 理解 Java 关键字 super 和关键字 this

大纲

接下来…

包

继承

访问控制

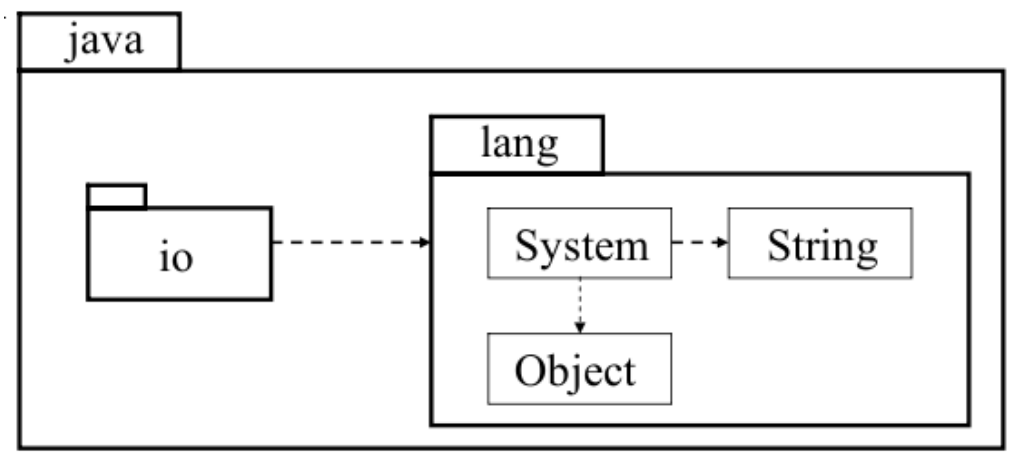
方法重写

关键字 super

关键字 this

什么是包？

为便于管理大型软件系统中数目众多的类，解决类的命名冲突问题以及进行访问控制，Java 引入包（package）机制，即将若干功能相关的类逻辑上分组打包到一起，提供类的多重类命名空间。



JDK API 中的常用包

包名	功能说明	包的含义
java.lang	Java 语言程序设计的基础类	language 的简写
java.awt	创建图形用户界面和绘制图形图像的相关类	抽象窗口工具集
java.util	集合、日期、国际化、各种实用工具	utility 的简写
java.io	可提供数据输入/输出相关功能的类	input/output 的简写
java.net	Java 网络编程的相关功能类	网络
java.sql	提供数据库操作的相关功能类	结构化查询语言的简写

包的创建

package 语句作为 Java 源文件的第一条语句，指明该文件中定义的类所在的包（若缺省该语句，则指定为无名包）。语法格式如下：

```
1 package pkg1[.pkg2[.pkg3...]];
```

CODE ▸ 创建包

```
1 package p1;
2 public class Test{
3   public void m1(){
4     System.out.println("In␣class␣Test,
5 ␣␣␣␣method␣m1␣is␣running!");
6   }
7 }
```

package 语句对所在源文件中定义的所有类型（包括接口、枚举、注解）均起作用。

包的创建

Java 编译器把包对应于文件系统的目录管理，package 语句中，用“.”来指明包（目录）的层次。如果在程序 Test.java 中已定义了包 p1，编译时采用如下方式：

```
1   > javac Test.java
```

则编译器会在当前目录下生成 Test.class 文件。
若在命令行下使用如下命令：

```
1   > java -d /home/xiaodong/work01 Test.java
```

“-d /home/xiaodong/work01”是传给 Java 编译器的参数，用于指定此次编译生成的.class 文件保存到该指定路径下，并且如果源文件中有 package 语句，则编译时会自动在目标路径下创建与包同名的目录 p1，再将生成的 Test.class 文件保存到该目录下。

导入包中的类

为使用定义在不同包中的 Java 类，需用 import 语句来引入所需要的类。语法格式：

```
1 import pkg1[.pkg2...].(classname|*);
```

CODE ◗ 导入和使用有名包中的类

```
1 import p1.Test; //or import p1.*;
2 public class TestPackage{
3   public static void main(String args[]){
4     Test t = new Test();
5     t.m1();
6   }
7 }
```

Java 包特性

一个类如果未声明为 public 的，则只能在其所在包中被使用，其他包中的类即使在源文件中使用 import 语句也无法引入它。可以不在源文件开头使用 import 语句导入要使用的有名包中的类，而是在程序代码中每次用到该类时都给出其完整的包层次，例如：

```
1  public class TestPackage{
2    public static void main(String args[]){
3      p1.Test t = new p1.Test();
4      t.m1();
5    }
6  }
```

接下来…

- 包
- 继承
- 访问控制
- 方法重写
- 关键字 super
- 关键字 this

什么是继承？

继承（Inheritance）是面向对象编程的核心机制之一，其本质是在已有类型基础之上进行扩充或改造，得到新的数据类型，以满足新的需要。

根据需要定义 Java 类描述“人”和“学生”信息：

CODE ▸ Class Person

```
1  public class Person {
2    public String name;
3    public int age;
4    public Date birthDate;
5    public String getInfo() {...}
6  }
```

什么是继承

CODE ◗ Class Student

```
1  public class Student {
2    public String name;
3    public int age;
4    public Date birthDate;
5    public String school;
6    public String getInfo() {...}
7  }
```

通过继承，简化 Student 类的定义：

CODE ◗ Class Student extends Person

```
1  public class Student extends Person {
2    public String school;
3  }
```

继承

Java 类（继承其他类）声明语法格式：

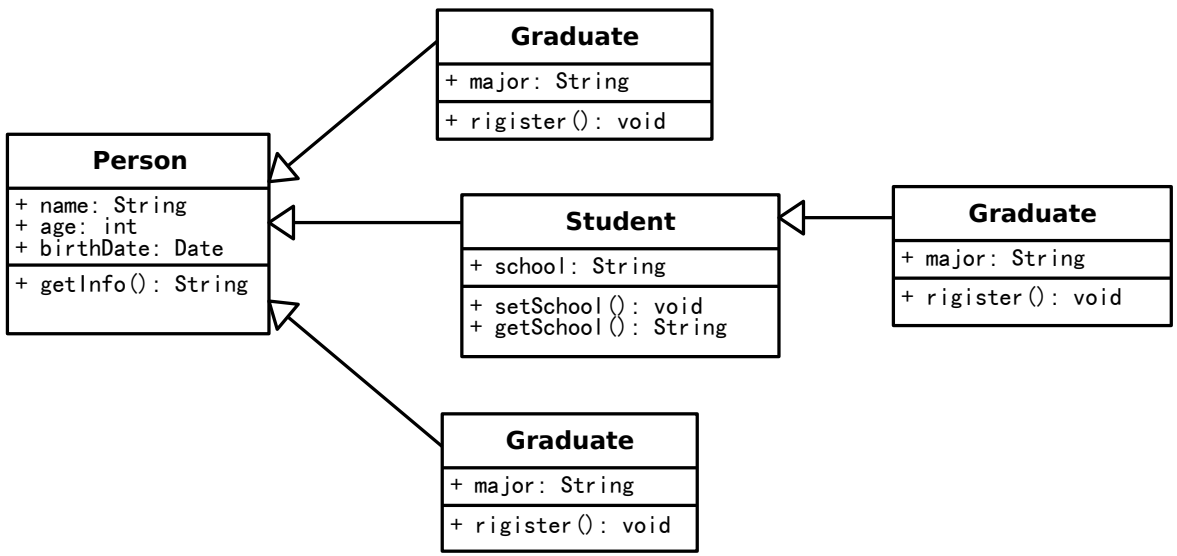
```
1  [< 修饰符 >] class < 类名 > [extends < 父类名 >] {
2    [< 属性声明 >]
3    [< 构造方法声明 >]
4    [< 方法声明 >]
5  }
```

Object 类是所有 Java 类的最高层父类, 如果在类的声明中未使用 extends 关键字指明其父类，则默认父类为 Object 类。

Java 继承的特性

Java 只支持单继承，不允许多重继承。

- 一个子类只能有一个父类；
- 一个父类可以派生出多个子类。



类之间的关系

依赖关系 一个类的方法中使用到另一个类的对象（uses-a）[1]。

聚合关系 一个类的对象包含（通过属性引用）了另一个类的对象（has-a）[2]。

泛化关系 一般化关系（is-a），表示类之间的继承关系、类和接口之间的实现关系以及接口之间的继承关系。

[1] 车能够装载货物，车的装载功能（load() 方法）对货物（goods）有依赖。
[2] 车有发动机、车轮等，Car 对象是由 Engine 等对象构成的。

接下来…

- 包
- 继承
- 访问控制
- 方法重写
- 关键字 super
- 关键字 this

访问控制

访问控制是指对 Java 类或类中成员的操作进行限制，即规定其在多大的范围内可以被直接访问。

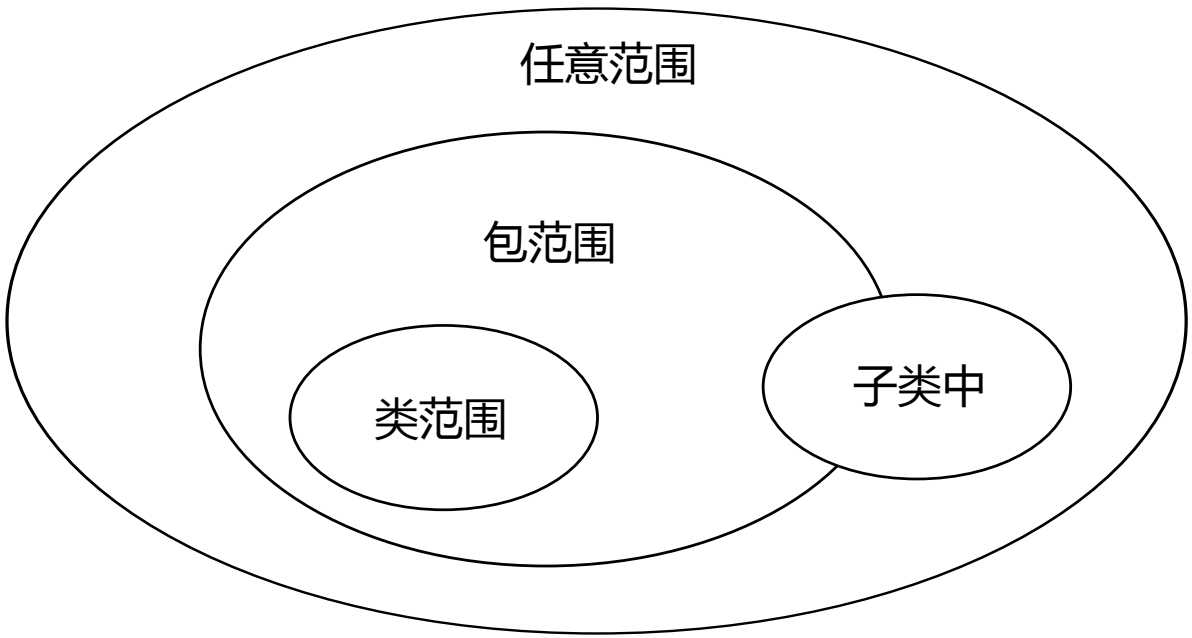
❖ 类的访问控制

在声明 Java 类时可以在 class 关键字前使用 public 来修饰，也可以不使用该修饰符。public 的类可在任意场合被引入和使用，而非 public 的类只能在其所在包中被使用。

❖ 类中成员的访问控制

修饰符/作用范围	同一个类	同一个包	子类	任何地方
public	yes	yes	yes	yes
protected	yes	yes	yes	no
无修饰符	yes	yes	no	no
private	yes	no	no	no

访问控制



访问控制注意的一些问题

- 一般不提倡将属性声明为 public 的，而构造方法和需要外界直接调用的普通方法则适合声明为 public 的。
- 在位于不同的包内，必须是子类的对象才可以直接访问其父类的 protected 成员，而父类自身的对象反而不能访问其所在类中声明的 protected 成员。
- 所谓“访问控制”只是控制对 Java 类或类中成员的直接访问，而间接访问是不做控制的，也不该进行控制。

访问控制 protected

CODE ⏵ A.java

```
1  package p1;
2  public class A {
3    public int m = 5;
4    protected int n = 6;
5  }
```

CODE ⏵ B.java

```
1  package p2;
2  import p1.A;
3  public class B extends A{
4    public void mb() {
5      m = m + 1;
6      n = n * 2;
7    }
8    public static void main(String[] args) {
9      B b = new B();
10     b.m = 7; // 合法
11     b.n = 8; // 合法
12     A a = new A();
13     a.m = 9 // 合法
14     a.n = 10 // 非法
15   }
16 }
```

接下来…

- 包
- 继承
- 访问控制
- 方法重写
- 关键字 super
- 关键字 this

方法重写

❖ 什么是方法重写

在子类中可以根据需要对从父类中继承来的方法进行重新定义，此称方法重写（Override）或覆盖。

❖ 语法规则

- 重写方法必须和被重写方法具有相同的方法名称、参数列表和返回值类型；
- 重写方法不能使用比被重写方法更严格的访问权限；
- 重写方法不允许声明抛出比被重写方法范围更大的异常类型。

方法重写示例

CODE 方法重写示例 A

```
1  public class Person {
2    String name;
3    int age;
4    public String getInfo() {
5      return "Name:"+ name + "\t" +"age:"+ age;
6    }
7  }
```

```
1  public class Student extends Person {
2    private String school;
3    public void setSchool(String scholl) {
4      this.school = school;
5    }
6    public String getSchool(){
7      return school;
8    }
9    public String getInfo() {
10     return "Name:"+ name + "\tAge:"+ age + "\tSchool:" + school;
11   }
12 }
```

方法重写示例

CODE ◗ 方法重写示例 B

```
1 public class Parent {
2   public void method1() {...}
3 }
```

```
1 public class Child extends Parent {
2   private void method1() {} //非法，权限更严格
3 }
```

```
1 public class UseBoth {
2   public void doOtherThing() {
3     Parent p1 = new Parent();
4     Child p2 = new Child();
5     p1.method1();
6     p2.method1();
7   }
8 }
```

同名属性

CODE ◗ 同名属性

```
1  public class Person {
2    int age = 5;
3    public int getAge() {
4      return age;
5    }
6    public int getInfo() {
7      return age;
8    }
9  }
```

```
1  public class Student extends Person {
2    int age = 6;
3    public int getAge() {
4      return age;
5    }
6  }
```

同名属性

```
1  public class Test {
2    public static void main(String args[]) {
3      Person p = new Person();
4      System.out.println(p.getAge());
5      Student s = new Student();
6      System.out.println(s.age);
7      System.out.println(s.getAge());
8      System.out.println(s.getInfo());
9    }
10 }
```

输出结果：

output

```
5
6
6
5
```

同名属性

❖ 对上述 Student 对象同名属性的几点说明

1. 以“对象名. 属性名”方式直接访问时，使用的是子类中添加的属性 age；
2. 调用子类添加或者重写的方法时，方法中使用的是子类定义的属性 age；
3. 调用父类中定义的方法时，方法中使用的是父类中的属性 age。

可以理解为“层次优先”[3]；不提倡使用同名属性。

[3]在哪个层次中的代码，就优先使用该层次类中定义的属性。

接下来…

- 包
- 继承
- 访问控制
- 方法重写
- 关键字 super
- 关键字 this

关键字 super

在存在命名冲突（子类中存在方法重写或添加同名属性）的情况下，子类中的代码将自动使用子类中的同名属性或重写后的方法。当然也可以在子类中使用关键字 super 引用父类中的成分：

访问父类中定义的属性

super.< 属性名 >

调用父类中定义的成员方法

super.< 方法名 >(< 实参列表 >)

子类构造方法中调用父类的构造方法

super(< 实参列表 >)

super 的追溯不仅限于直接父类，先从直接父类开始查找，如果找不到则逐层上溯，一旦在某个层次父类中找到匹配成员即停止追溯并使用该成员。

super 用法示例

CODE ◗ super A

```
1   class Animal {
2     protected int i = 1; //用于测试同名属性，无现实含义
3   }

5   class Person extends Animal {
6     protected int i = 2; //用于测试同名属性，无现实含义
7     private String name = "Tom";
8     private int age = 9;
9     public String getInfo() {
10      return "Name:" + name + "\tAge:" + age;
11    }
12    public void testI() {
13      System.out.println(super.i);
14      System.out.println(i);
15    }
16  }
```

super 用法示例

CODE ◗ super B

```
1   class Student extends Person {
2     private int i = 3;
3     private String school = "THU";
4     public String getInfo() {  //重写方法
5       return super.getInfo() + "\tSchool:" + school;
6     }
7     public void testI() {   //重写方法
8       System.out.println(super.i);
9       System.out.println(i);
10    }
11  }
12  public class Test {
13    public static void main(String args[]) {
14      Person p = new Person();
15      System.out.println(p.getInfo());
16      p.testI();
17      Student s = new Student();
18      System.out.println(s.getInfo());
19      s.testI();
20    }
21  }
```

super 用法示例

上述代码的输出结果：

output

```
Name:Tom Age:9
1
2
Name:Tom Age:9 School:THU
2
3
```

接下来…

- 包
- 继承
- 访问控制
- 方法重写
- 关键字 super
- 关键字 this

关键字 this

在 Java 方法中，不但可以直接使用方法的局部变量，也可以使用调用该方法的对象。
为解决可能出现的命名冲突，Java 语言引入 this 关键字来标明方法的当前对象。分为两种情况：

- 在普通方法中，关键字 this 代表方法的调用者，即本次调用了该方法的对象；
- 在构造方法中，关键字 this 代表该方法本次运行所创建的那个新对象。

this 作为一个特殊的引用类型变量，可以通过“this. 成员”的方式访问其引用的当前对象的属性和方法。

关键字 this

CODE ▶ this 用法示例

```
1  public class MyDate {
2    private int day = 17;
3    private int month = 2;
4
5    public MyDate(int day, int month) {
6      this.day = day; // A
7      this.month = month;
8    }
9    ... // Some methods
10
11   public void setAll(int day, int month) {
12     this.setMonth(month); // B
13     this.setDay(day);
14   }
15 }
```

关键字 this

❖ 关于 this 的归纳说明

1. 在 Java 方法中直接给出变量名而不是“对象名. 变量名”的方式访问一个变量，系统首先尝试作为局部变量来处理；如果方法中不存在该名字的局部变量，才会到方法当前对象的成员变量中查找。
2. 在 Java 方法中直接调用一个方法而不指定其调用者时，则默认调用者为当前对象 this。

本节习题

1. 类之间具备哪些关系？
2. 比较说明 super 和 this 的含义。

THE END

wangxiaodong@ouc.edu.cn