Java 程序设计

第6章 接口和内部类

田英鑫 tyx@hit.edu.cn
哈尔滨工业大学软件学院

第6章 接口和内部类

& 本章导读

- 6.1 接口
- 6.2 对象克隆
- 6.3 接口与回调
- 6.4 内部类

第6章 接口和内部类

& 本章重点

- 6.1 接口
- 6.4 内部类

& 本章难点

- 6.3 接口与回调
- 6.2 对象克隆
- 6.4 内部类
  - 匿名内部类
  - 静态内部类

6.1 接口

- 什么是接口？
  - 接口是一种与类相似的结构，但接口不是类，而是一组对类的要求，这些类要与接口一致
  - 一个类可以实现一个或多个接口，并在需要接口的地方，随时使用实现了相应接口的对象
- 接口的定义和使用

```
interface NameOfInterface
{
   ...
}
class NameOfClass implements NameofInterface
{
   ...
}
```

6.1 接口

- 举例：Comparable 接口
  - Arrays类中的sort方法承诺可以对对象数组进行排序，但要求满足下列前提：
    - 对象所属的类必须实现Comparable接口

```
public interface Comparable
{
    int CompareTo(Object other);//该方法没有实现
}
class Employee implements Comparable
{
    public int compareTo(Object otherObject)
    {
        ...
    }
}
```

6.1 接口

- 接口的实现
  - 实现接口的类中必须对接口中的方法进行定义，否则该类就是一个抽象类，不能实例化对象
  - 例如，在Employee类中需要对compareTo方法进行定义

```
public int compareTo(Object otherObject)
{
   Employee other = (Employee)otherObject;
   if(salary < other.salary)
      return -1;
   if(salary > other.salary)
      return 1;
   return 0;
}
```

6.1 接口

JDK 1.5 中的 Comparable 接口

```
public interface Comparable<T>
{
   int CompareTo(T other);
}
class Employee implements Comparable<Employee>
{
   public int compareTo(Employee other)
   {
      if(salary < other.salary)
         return -1;
      if(salary > other.salary)
         return 1;
      return 0;
   }
}
```

参见教材例6-1:EmployeeSortTest.java

6.1 接口

- 接口的特性
  - 接口不是类，不能使用new操作符实例化接口

```
x = new Comparable(...); //错误，不能创建接口对象
```

  - 可以声明接口变量，该变量必须指向一个实现了该接口的类的对象

```
Comparable x;
x = new Employee(...);
    //Employee类实现了Comparable接口
```

  - 可以使用instanceof操作符判断对象是否实现了某个接口

```
if(anObject instanceof Comparable) {...}
```

6.1 接口

- 接口的特性
  - 与可以建立类的继承关系一样，一个接口也可以扩展自另一个接口
  - 接口中不能包含实例字段和静态方法，但接口中可以声明常量
    - 接口中的方法自动被设置为 public
    - 字段被自动设置成 public static final
  - 每个类只能有一个超类，但类能够实现多个接口
    - 例如：Employee类实现了Cloneable和Comparable两个接口，多个接口之间用逗号分隔

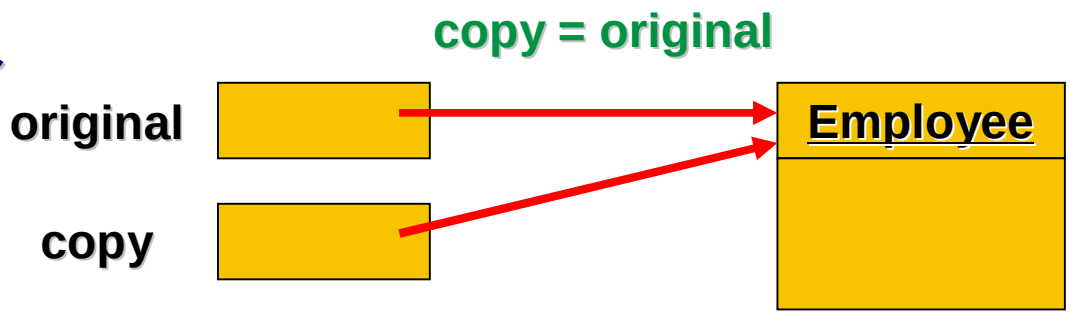
      class Employee implements Cloneable, Comparable

6.1 接口

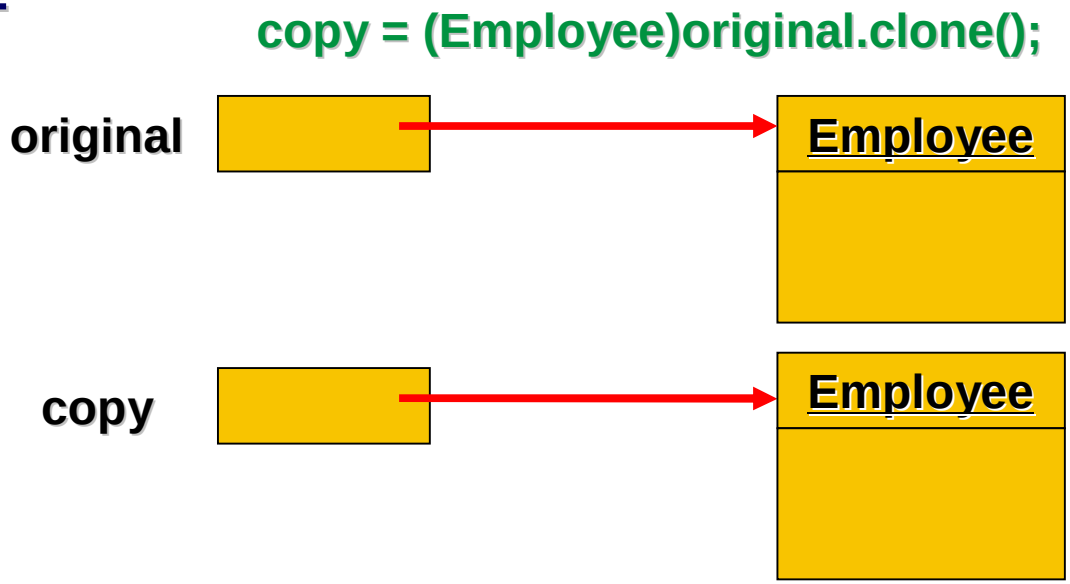
- 接口和抽象类
  - 抽象类中可以含有非抽象方法，而接口中的方法都是抽象的
  - 一个类只能派生自一个抽象类，但可以同时实现多个接口，以达到多重继承（多种约束）的目的
- 一些疑问？
  - 接口中能有构造方法吗？
  - 接口能继承（extends）另一个接口，但接口能实现（implements）另一个或多个接口吗？

6.2 对象克隆

- 拷贝


copy = original
original
copy
Employee

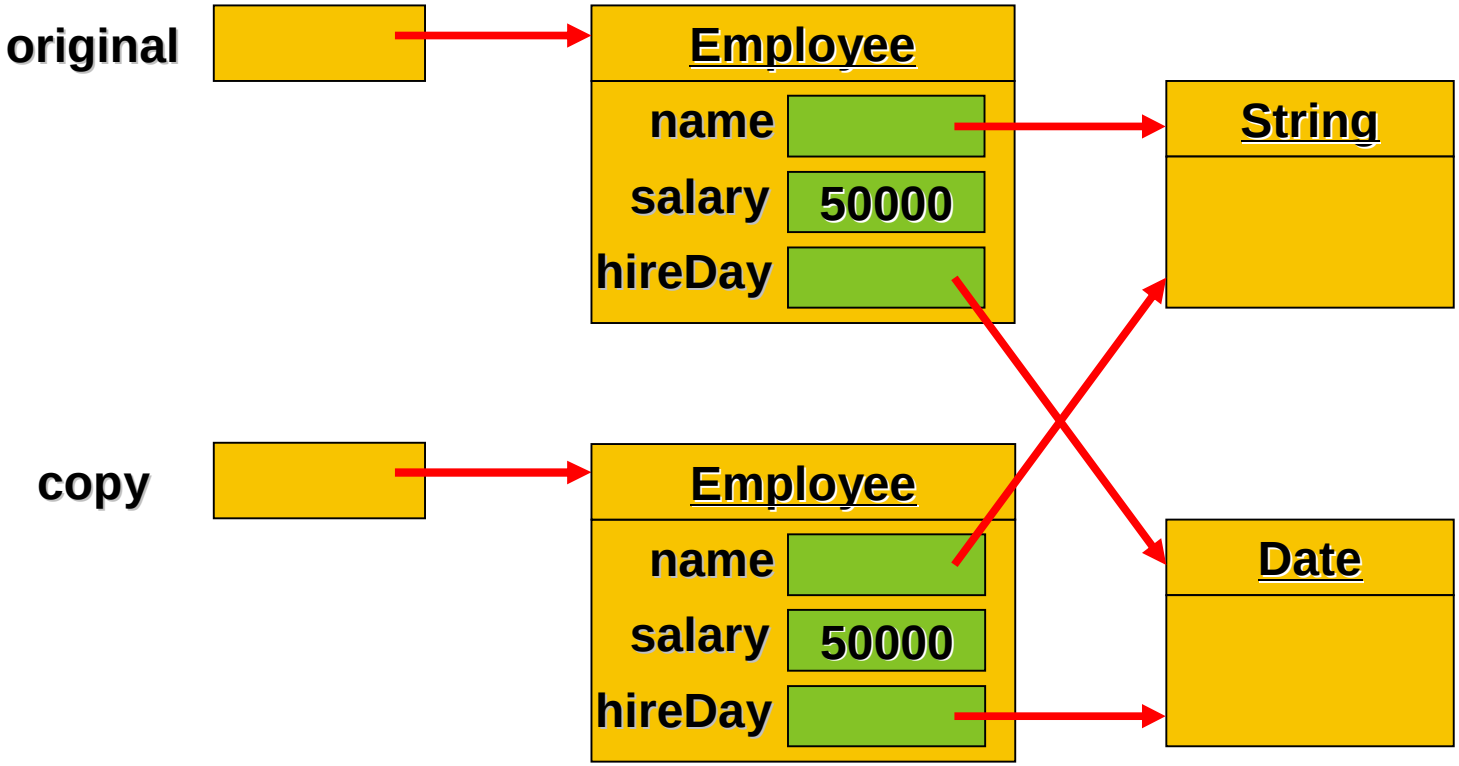
- 克隆


copy = (Employee)original.clone();
original
copy
Employee
Employee

6.2 对象克隆

- 浅拷贝

```
copy = (Employee)original.clone();
```



6.2 对象克隆

- 实现 Cloneable 接口
  - 类的clone方法继承自Object类，clone方法返回Object类型，使用clone方法时必须对结果进行强制转换
  - 要对一个类的对象使用clone方法，该类必须实现Cloneable标记接口，否则将抛出异常

    CloneNotSupportedException

6.2 对象克隆

- 深拷贝
  - 重写Object类的clone方法

```
class Employee implements Cloneable {
    ...
    public Object clone() {
        try {
            Employee cloned = (Employee)super.clone();
            cloned.hireDay= (Date)hireDay.clone();
            return cloned;
        }
        catch(CloneNotSupportedException e) {
            return null;
        }
    }
}
```

参见教材例6-2:CloneTest.java

6.3 接口与回调

- 回调
  - 回调模式是一种常见的编程模式，如C/C++中的回调函数通过将回调函数的地址传给调用者从而实现调用
  - 回调技术的一个典型应用是用于事件处理
  - Java中使用接口可以实现回调模式

```
class TimerPrinter implements ActionListener
{
   public void actionPerformed(ActionEvent event)
   {
      //do something...
   }
}
```

参见教材例6-3:TimerTest.java

6.4 内部类

- 什么是内部类？
  - 内部类是定义在另一个类中的类
- 使用内部类的原因
  - 内部类方法可以访问该类定义所在的作用域中的数据，包括私有数据
  - 内部类可以被同一个包中的其他类隐藏起来
  - 当想要定义一个回调函数且不想编写大量代码时，使用匿名内部类比较便捷

6.4 内部类

n 内部类的定义

```
class TalkingClock
{
    public TalkingClock(int interval,boolean beep) {
        ...
    }
    public void start() {
        ...
    }
    private int interval;
    private boolean beep;

    private class TimePrinter implements ActionListener {
        ...
    }
}
```

6.4 内部类

- 使用内部类访问对象状态
  - 内部类可以访问外部类的数据

```
private class TimePrinter implements ActionListener
{
   public void actionPerformed(ActionEvent event)
   {
      Date now = new Date();
      System.out.println("At the tone,the time is " + now);
      if(beep)
         Tookit.getDefaultToolkit().beep();
   }
}
```

参见教材例6-4:InnerClassTest.java

6.4 内部类

局部内部类

```
public void start(double rate) {
   class InterestAdder implements ActionListener {
      public InterestAdder(double aRate) {rate = aRate;}
      public void actionPerformed(ActionEvent event) {
         double interest = balance * rate / 100;
         balance += interest;
         NumberFormat formatter =
            NumberFormat.getCurrencyInstance();
         System.out.println("balance=" +
            formatter.format(balance));
      }
      private double rate;
   }
   ActionListener adder = new InterestAdder(rate);
   Timer t = new Timer(1000,adder); t.start();
}
```

6.4 内部类

n 匿名内部类

```
public void start(double rate) {
   ActionListener adder = new
      ActionListener() {
         public void actionPerformed(ActionEvent event) {
            double interest = balance * rate / 100;
            balance += interest;
            NumberFormat formatter =
               NumberFormat.getCurrencyInstance();
            System.out.println("balance=" +
               formatter.format(balance));
         }
   };
   Timer t = new Timer(1000,adder);
   t.start();
}
```

参见教材例6-5:AnonymousInnerClassTest.java

6.4 内部类

- 静态内部类
  - 当内部类不需要有对外部类对象的引用时，内部类可以声明为static的

```
class ArrayAlg
{
  public static class Pair
  {
    ...
  }
  ...
}
```

参见教材例6-6:StaticInnerClassTest.java

Any Question?