Java 多态 Java 封装

Java 抽象类

在面向对象的概念中，所有的对象都是通过类来描绘的，但是反过来，并不是所有的类都是用来描绘对象的，如果一个类中没有包含足够的信息来描绘一个具体的对象，这样的类就是抽象类。

抽象类除了不能实例化对象之外，类的其它功能依然存在，成员变量、成员方法和构造方法的访问方式和普通类一样。

由于抽象类不能实例化对象，所以抽象类必须被继承，才能被使用。也是因为这个原因，通常在设计阶段决定要不要设计抽象类。

父类包含了子类集合的常见的方法，但是由于父类本身是抽象的，所以不能使用这些方法。

在Java中抽象类表示的是一种继承关系，一个类只能继承一个抽象类，而一个类却可以实现多个接口。

抽象类

在Java语言中使用abstract class来定义抽象类。如下实例：

Employee.java 文件代码：

```
/* 文件名 : Employee.java */
public abstract class Employee
{
private String name;
private String address;
private int number;
public Employee(String name, String address, int number)
{
System.out.println("Constructing an Employee");
this.name = name;
this.address = address;
this.number = number;
}
public double computePay()
{
System.out.println("Inside Employee computePay");
return 0.0;
}
public void mailCheck()
{
System.out.println("Mailing a check to " + this.name
+ " " + this.address);
}
public String toString()
{
return name + " " + address + " " + number;
}
public String getName()
{
return name;
}
```

```
public String getAddress()
{
return address;
}
public void setAddress(String newAddress)
{
address = newAddress;
}
public int getNumber()
{
return number;
}
}
```

注意到该 Employee 类没有什么不同，尽管该类是抽象类，但是它仍然有 3 个成员变量，7 个成员方法和 1 个构造方法。 现在如果你尝试如下的例子：

AbstractDemo.java 文件代码：

```
/* 文件名 : AbstractDemo.java */
public class AbstractDemo
{
public static void main(String [] args)
{
/* 以下是不允许的，会引发错误 */
Employee e = new Employee("George W.", "Houston, TX", 43);
System.out.println("\n Call mailCheck using Employee reference--");
e.mailCheck();
}
}
```

当你尝试编译AbstractDemo类时，会产生如下错误：

```
Employee.java:46: Employee is abstract; cannot be instantiated
      Employee e = new Employee("George W.", "Houston, TX", 43);
                   ^
1 error
```

继承抽象类

我们能通过一般的方法继承Employee类：

Salary.java 文件代码：

```
/* 文件名 : Salary.java */
public class Salary extends Employee
{
private double salary; //Annual salary
public Salary(String name, String address, int number, double
salary)
{
super(name, address, number);
```

```
setSalary(salary);
}
public void mailCheck()
{
System.out.println("Within mailCheck of Salary class ");
System.out.println("Mailing check to " + getName()
+ " with salary " + salary);
}
public double getSalary()
{
return salary;
}
public void setSalary(double newSalary)
{
if(newSalary >= 0.0)
{
salary = newSalary;
}
}
public double computePay()
{
System.out.println("Computing salary pay for " + getName());
return salary/52;
}
}
```

尽管我们不能实例化一个 Employee 类的对象，但是如果我们实例化一个 Salary 类对象，该对象将从 Employee 类继承 7 个成员方法，且通过该方法可以设置或获取三个成员变量。

AbstractDemo.java 文件代码：

```
/* 文件名 : AbstractDemo.java */
public class AbstractDemo
{
public static void main(String [] args)
{
Salary s = new Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00);
Employee e = new Salary("John Adams", "Boston, MA", 2, 2400.00);
System.out.println("Call mailCheck using Salary reference --");
s.mailCheck();
System.out.println("\n Call mailCheck using Employee reference--");
e.mailCheck();
}
}
```

以上程序编译运行结果如下：

```
Constructing an Employee
Constructing an Employee
Call mailCheck using  Salary reference --
Within mailCheck of Salary class
Mailing check to Mohd Mohtashim with salary 3600.0
```

```
Call mailCheck using Employee reference--
Within mailCheck of Salary class
Mailing check to John Adams with salary 2400.
```

抽象方法

如果你想设计这样一个类，该类包含一个特别的成员方法，该方法的具体实现由它的子类确定，那么你可以在父类中声明该方法为抽象方法。

Abstract 关键字同样可以用来声明抽象方法，抽象方法只包含一个方法名，而没有方法体。

抽象方法没有定义，方法名后面直接跟一个分号，而不是花括号。

```
public abstract class Employee
{
private String name;
private String address;
private int number;
public abstract double computePay();
//其余代码
}
```

声明抽象方法会造成以下两个结果：

- 如果一个类包含抽象方法，那么该类必须是抽象类。
- 任何子类必须重写父类的抽象方法，或者声明自身为抽象类。

继承抽象方法的子类必须重写该方法。否则，该子类也必须声明为抽象类。最终，必须有子类实现该抽象方法，否则，从最初的父类到最终的子类都不能用来实例化对象。

如果Salary类继承了Employee类，那么它必须实现computePay()方法：

Salary.java 文件代码：

```
/* 文件名 : Salary.java */
public class Salary extends Employee
{
private double salary; // Annual salary
public double computePay()
{
System.out.println("Computing salary pay for " + getName());
return salary/52;
}
//其余代码
}
```

抽象类总结规定

- 1. 抽象类不能被实例化(初学者很容易犯的错)，如果被实例化，就会报错，编译无法通过。只有抽象类的非抽象子类可以创建对象。

- 2. 抽象类中不一定包含抽象方法，但是有抽象方法的类必定是抽象类。
- 3. 抽象类中的抽象方法只是声明，不包含方法体，就是不给出方法的具体实现也就是方法的具体功能。
- 4. 构造方法，类方法（用 static 修饰的方法）不能声明为抽象方法。
- 5. 抽象类的子类必须给出抽象类中的抽象方法的具体实现，除非该子类也是抽象类。

Java 多态 Java 封装