【济南中心】JAVA 编程阶梯：基础篇之第九章

- **面向对象之多态**

多态(polymorphic)概述：事物存在的多种形态

体现：父类的引用或者接口的引用指向了自己的子类对象。

多态前提：

* a:要有继承关系。

* b:要有方法重写。

* c:要有父类引用指向子类对象。

```
class Fu {
		public void show() {
			System.out.println("fu show");
		}
}

class Zi extends Fu {
		public void show() {
			System.out.println("zi show");
		}
}

class Demo {
		public static void main(String[] args) {
			Fu f = new Zi();// 父类的引用或者接口的引用指向了自己的子类对象
		}
}
```

- **多态中的成员访问特点**

成员变量：

当子父类中出现同名的成员变量时。

编译时期：参考的是引用型变量所属的类中是否有被调用的成员变量。没

有，编译失败。

运行时期：也是调用引用型变量所属的类中的成员变量。

简单记：编译和运行都参考等号的左边。

成员函数：

编译时期：参考左边，如果没有，编译失败。

运行时期：参考右边的对象所属的类。

编译看左边，运行看右边。

对于成员函数是动态绑定到对象上。

静态函数。

静态函数是静态的绑定到类上。

编译和运行都参考左边。

【结论】

对于成员变量和静态函数，编译和运行都看左边。

对于成员函数，编译看左边，运行看右边。

```
class Fu {
		int num = 10;

		public void show() {
				System.out.println("fu show" + num);
		}
}

class Zi extends Fu {
		int num = 20;
		int m = 30;

		public void show() {
				System.out.println("zi show " + num);
		}
```

```
		public void method() {
			System.out.println("zi method");
		}
}

class Demo {
		public static void main(String[] args) {
			Fu f = new Zi();
			// f.method(); 成员方法编译看左边，父类中没有，
所以编译失败
			f.show();// 运行看右边。
			// int sum = f.m; //成员变量编译和运行都参考等号
的左边，父类中没有编译失败
			int sum = f.num;
			System.out.println(sum);// 成员变量编译和运行都
参考等号的左边，所以是父类中的成员变量
		}
}
```

- **抽象函数**

  抽象类概述：

  抽象就是看不懂的

  抽象类特点：

  a:抽象类和抽象方法必须用 abstract 关键字修饰

  abstract class 类名 {}

  public abstract void eat();

  b:抽象类不一定有抽象方法，有抽象方法的类一定是抽象类或者是

  接口

  c:抽象类不能实例化那么，可以按照多态的方式，由具体的子类实

  例化。其实这也是多态的一种，抽象类多态。

  d:抽象类的子类要么是抽象类要么重写抽象类中的所有抽象方法

```
abstract class People //extends Object
	{
```

```
abstract void work();//抽象函数。需要 abstract 修饰，并分号;结束
        }
```

- **抽象类的成员特点**

成员变量：既可以是变量，也可以是常量。

构造方法：抽象类是一个类，所以，它有构造方法，用于子类访问父类数据的初始化。

成员方法：既可以是抽象的，也可以是非抽象的。

```
abstract class People {

                public People() {
                                //构造方法
                }

                String name = "小明";// 变量
                final int GENDE = 1;// 常量

                abstract void work();// 抽象方法

                public void sleep() {// 非抽象方法
                                System.out.println("睡觉.....");
                }
}
```

抽象类的成员方法特性：

a:抽象方法 强制要求子类做的事情。

b:非抽象方法 子类继承的事情，提高代码复用性。

```
abstract class People // extends Object
{
                  abstract void work();// 抽象函数。需要 abstract 修饰，并分号;
结束
}

class Worker extends People {

                @Override
```

```
                void work() {
                        System.out.println("工作中.....");

                }
}

class Demo {
                public static void main(String[] args) {
                        People p = new Worker();
                        p.work();
                }
}
```

抽象类和一般类的异同点：

相同：

1、它们都是用来描述事物的。

2、它们之中都可以定义属性和行为。

不同：

1、一般类可以具体的描述事物。抽象类描述事物的信息不具体

2、抽象类中可以多定义一个成员：抽象函数。

3、一般类可以创建对象，而抽象类不能创建对象。

抽象类中是可以不定义抽象方法，不让该类创建对象，不让别的类建立该抽象类对象。

抽象关键字 abstract 不可以和 final，private，static 共存。

private:

私有内容子类继承不到，所以，不能重写。

但是 abstract 修饰的方法，要求被重写。两者冲突。

final

final 修饰的方法不能被重写。

而 abstract 修饰的方法，要求被重写。两者冲突。

static

假如一个抽象方法能通过 static 修饰，那么这个方法，就可以直接通过类名调用。

而抽象方法是没有方法体的，这样的调用无意义。所以，不能用 static 修饰。

- **接口**

接口概述

从狭义的角度讲就是指 java 中的 interface

从广义的角度讲对外提供规则的都是接口

接口特点

a:接口用关键字 interface 表示

interface 接口名 {}

b:类实现接口用 implements 表示

class 类名 implements 接口名 {}

c:接口不能实例化

可以按照多态的方式来实例化。

d:接口的子类

a:可以是抽象类。但是意义不大。

b:可以是具体类。要重写接口中的所有抽象方法。(推荐方案)

```
interface People {
		abstract void work();
}

class Worker implements People {

		@Override
		public void work() {
			System.out.println("工作中...");

		}

}

class Demo {
		public static void main(String[] args) {
			People p = new Worker();
			p.work();
		}
}
```

- **接口的成员特点**

  接口成员特点
  成员变量：只能是常量，并且是静态的并公共的。
  默认修饰符：public staticfinal
  构造方法：接口没有构造方法。
  成员方法：只能是抽象方法。
  默认修饰符：public abstract

- **类与类,类与接口,接口与接口的关系**

  类与类,类与接口,接口与接口的关系

  类与类：

  继承关系,只能单继承,可以多层继承。

  类与接口：

  实现关系,可以单实现,也可以多实现。

  并且还可以在继承一个类的同时实现多个接口。

接口与接口：

继承关系,可以单继承,也可以多继承。

- **抽象类和接口的区别**

成员区别

抽象类：

成员变量：可以变量，也可以常量

构造方法：有

成员方法：可以抽象，也可以非抽象

接口：

成员变量：只可以常量

成员方法：只可以抽象

关系区别：

类与类之间有继承关系，可以单继承不能多继承

类与接口之间有实现关系，可以单实现或者多实现

接口与接口之间有继承关系，可以单继承也可以多继承

设计理念区别：

抽象类 被继承体现的是：”is a”的关系。抽象类中定义的是该继承体系的共性功能。

接口 被实现体现的是：”like a”的关系。接口中定义的是该继承体系的扩展功能。

识别二维码 关注黑马程序员视频库
免费获得更多 IT 资源