方法重写 VS 方法重载

位置：方法重写在子类中对父类方法进行重写，方法重载是在同一个类中。

方法名：方法重写相同，方法重载相同。

参数列表：方法重写相同，方法重载不同。

返回值：方法重写相同或是其子类，方法重载没有要求。

访问权限：方法重写不能小于父类，方法重载没有要求。

多态

一个事物具有多种表现形态，在 Java 程序中，定义一个方法，在具体的生成环境中根据不同的需求呈现不同的业务逻辑，多态的前提是继承。

```
package com.southwind.test;

public class Memeber {
  public void buyBook() {

  }
}
```

```
package com.southwind.test;

public class OrdinaryMember extends Memeber {
  public void buyBook() {
    System.out.println("普通会员买书打9折");
  }
}
```

```
package com.southwind.test;

public class SuperMember extends Memeber {
  public void buyBook() {
    System.out.println("超级会员买书打6折");
  }
}
```

```
package com.southwind.test;

public class Cashier {
  private Memeber memeber;
```

```
  public Memeber getMemeber() {
    return memeber;
  }

  public void setMemeber(Memeber memeber) {
    this.memeber = memeber;
  }

  public void settlement() {
    this.memeber.buyBook();
  }
}
```

```
package com.southwind.test;

public class Test {
  public static void main(String[] args) {
    OrdinaryMember ordinaryMember = new OrdinaryMember();
    SuperMember superMember = new SuperMember();
    Cashier cashier = new Cashier();
    cashier.setMemeber(superMember);
    cashier.settlement();
  }
}
```

多态的具体使用有两种形式：

1、定义方法时形参类型为父类，实际调用方法时传入子类类型的参数。

2、定义方法时返回值类型为父类，实际调用方法时返回子类对象。

以上两种形式的基本原理都是父类引用可以指向子类对象。

```
public void settlement(Memeber memeber) {
  memeber.buyBook();
}

Cashier cashier = new Cashier();
OrdinaryMember ordinaryMember = new OrdinaryMember();
cashier.settlement(ordinaryMember);
```

```
public Member getMember(String name) {
  if(name.equals("ordinary")) {
    return new OrdinaryMember();
  }else {
    return new SuperMember();
  }
}
```

抽象方法和抽象类

如果一个方法只有方法的声明而没有具体的方法实现，这个方法就叫做抽象方法，Java 中的抽象方法需要使用 abstract 关键字来修饰。

```
public abstract void buyBook();
```

一旦类中定义了抽象方法，则该类也必须声明为抽象类，需要在类定义处添加 abstract 关键字。

```
public abstract class Member {
  public abstract void buyBook();
}
```

抽象类与普通类的区别是抽象类不能被实例化，抽象方法与普通方法的区别是抽象方法没有方法体。

抽象类中可以没有抽象方法，但是包含了抽象方法的类必须定义为抽象类。即我们可以在抽象类中定义普通方法，但是在普通类中不能定义抽象方法。

如果父类是抽象类，一旦子类继承了该抽象父类，则子类必须对父类的抽象方法进行重写，否则程序报错。

```
package com.southwind.test;

public abstract class Member {
  public abstract void buyBook();
}
```

```
package com.southwind.test;

public class SuperMember extends Member {
  @Override
  public void buyBook() {
    // TODO Auto-generated method stub
    System.out.println("超级会员买书打6折");
  }
}
```

如果子类也是抽象类，则可以不用重写父类的抽象方法。