上周课程小结

1. 接口的定义与使用
2. 接口回调
3. 面向接口编程
4. 内部类
5. 匿名类（与子类相关）

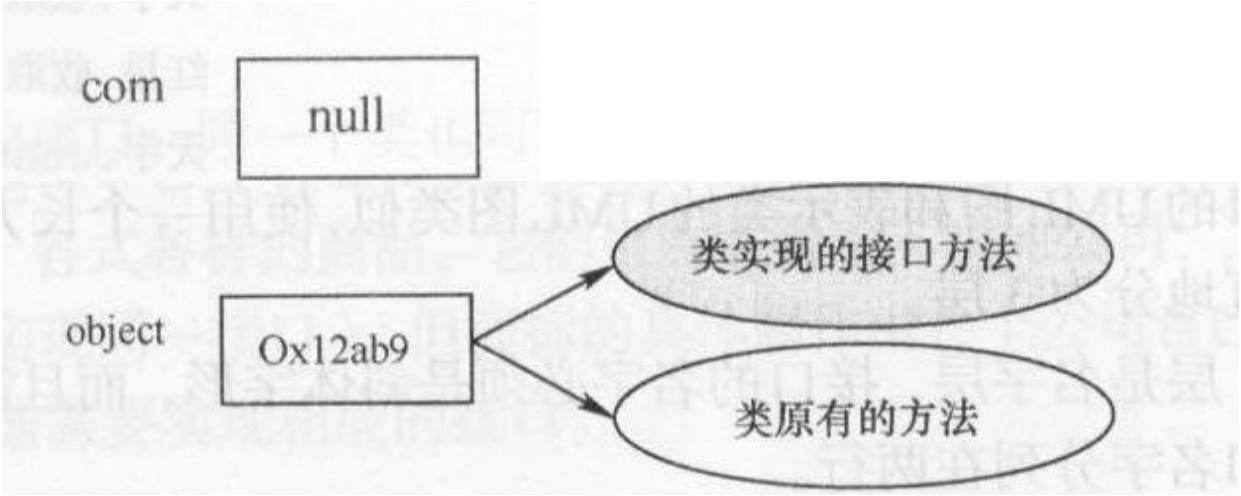
复习：接口回调

- 接口回调是指：把实现某一接口的类创建的对象的引用，赋给该接口声明的变量。
- 当接口变量调用被类实现的接口方法时，就是通知相应的对象调用这个方法。

复习：接口变量与回调机制

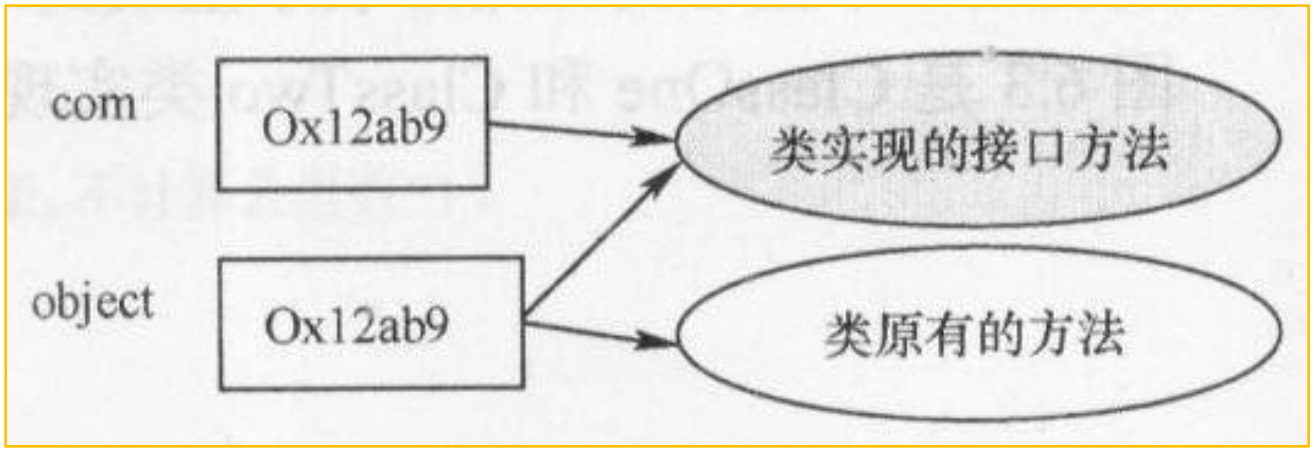
- 接口变量属于引用型变量，可存放实现该接口的类创建的对象引用。

```
Com com;
ImpleCom object = new ImpleCom();
```



复习：接口变量与回调机制

com=object;



- 接口回调非常类似于上转型对象调用子类的重写方法。
- 接口不能调用类中其它非接口方法。

复习：接口与abstract类的比较

（1）在表现形式上，接口和abstract类有何区别？

（2）在应用场景上，接口和abstract类有何区别？

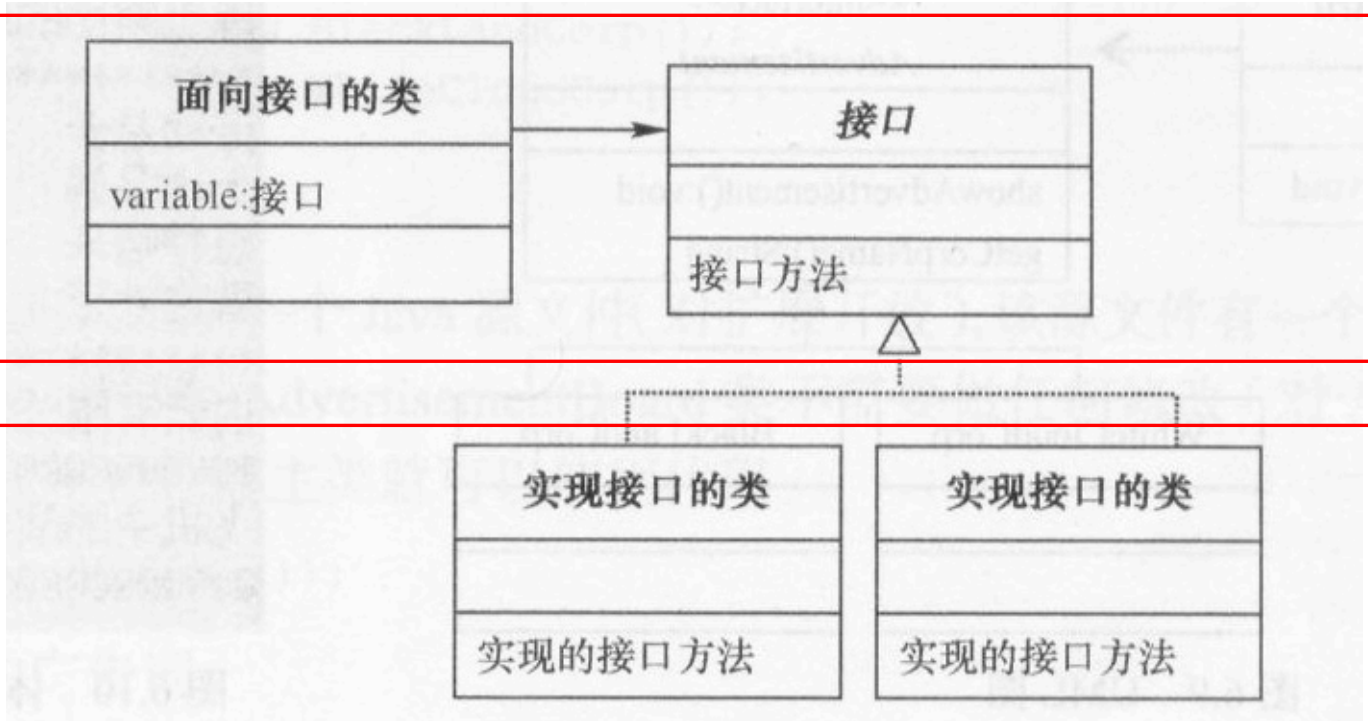
Eclipse演示

复习：面向接口编程

➢ 面向接口编程也可体现程序设计“开-闭”原则，即对扩展开放，对修改关闭。

闭

开



复习：和子类有关的匿名类

- 假如没有显式地声明一个类的子类，如何用子类创建一个对象？
- 使用子类的类体创建一个子类对象

创建子类对象时，除了使用父类的构造方法外还有子类类体，此类体被认为是一个子类去掉类声明后的类体，称作子类相关匿名类。

复习：和子类有关的匿名类

```java
package shu.ces.java.chap7;

public class ShowBoard {
  void showMess(OutputAlphabet show) {
    show.output();
  }
}
```

```java
package shu.ces.java.chap7;

abstract class OutputAlphabet {
  public abstract void output();
}
```

```java
package shu.ces.java.chap7;

public class OutputEnglish extends OutputAlphabet {
  public void output() {
    for(char c='a';c<='z';c++) {
      System.out.printf("%3c",c);
    }
  }
}
```

复习：和子类有关的匿名类

```
1  package shu.ces.java.chap7;
2
3  public class Example7_2 {
4    public static void main(String args[]) {
5      ShowBoard board=new ShowBoard();
6      board.showMess(new OutputEnglish());  //向参数传递OutputAlphabet的子类对象
7      board.showMess(new OutputAlphabet() {  //向参数传递OutputAlphabet的匿名子类对象
8                  public void output() {
9                     System.out.println();
10                    for(char c='α';c<='ω';c++)  //输出希腊字母
11                       System.out.printf("%3c",c);
12                  }
13                }
14               );
15   }
16 }
```

Eclipse演示

a b c d e f g h i j k l m n o p q r s t u v w x y z
α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ ς σ τ υ φ χ ψ ω

本周课程安排

1. 匿名类（与接口相关）
2. 异常类
3. 常用实用类
4. Java Swing概述
5. 窗口
6. 常用组件与布局（I）

2.2 与接口有关的匿名类

Java允许直接用接口名和一个类体创建一个匿名对象，此类体被认为是实现了该接口的类去掉类声明后的类体，也称作接口相关匿名类。

```
new Computable() {
        实现接口的匿名类的类体
}
```

如果某个方法的参数是接口类型，则可使用接口名和类体组合创建一个匿名对象传递给方法的参数。

2.2 与接口有关的匿名类

```
1  package shu.ces.java.chap7;
2
3  interface SpeakHello {
4       void speak();
5  }
6  class  HelloMachine {
7     public void turnOn(SpeakHello hello) {
8         hello.speak();
9     }
10 }
11 public class Example7_3 {
12    public static void main(String args[]) {
13       HelloMachine machine = new HelloMachine();
14       machine.turnOn( new SpeakHello(){
15                            public void speak() {
16                               System.out.println("hello,you are welcome!");
17                            }
18                         }
19                      );
20       machine.turnOn( new SpeakHello(){
21                            public void speak() {
22                                System.out.println("你好，欢迎光临！");
23                            }
24                         }
25                      );
26    }
27 }
28
```

Eclipse演示

3、异常类

- Java异常出现在方法调用过程中，即在方法调用过程中抛出异常对象，终止当前方法的继续执行，导致程序运行出现异常，进行异常处理。

3.1 try～catch语句

- Java使用try～catch语句处理异常。
- 将可能出现的异常操作放在try部分，当try部分中的某个方法调用发生异常后，try部分将立刻结束执行，而转向执行相应的catch部分，程序将发生异常后处理放在catch部分。
- try～catch语句可以由几个catch组成，分别处理发生的相应异常。
- finally{}在try～catch语句后，执行finally语句，也就是说，无论在try部分是否发生过异常，finally子语句都会被执行。

3.1 try～catch语句

- try～catch语句的格式如下：

```
try {
  包含可能发生异常的语句
}
catch(ExceptionSubClass1  e1) {
    …
}
catch(ExceptionSubClass2  e2) {
    …
}
Finally {
    …
}
```

- catch参数中的异常类都是Exception的某个子类，表明try部分可能发生的异常。这些子类之间不能有父子关系，否则保留一个含有父类参数的catch即可。

3.2 自定义异常类与异常处理

- 编写程序时可以扩展Exception类定义自己的异常类，然后根据程序的需要来规定哪些方法产生这样的异常。
- 一个方法在声明时可以使用throws关键字声明要产生的若干个异常；
- 在方法体中具体给出产生异常的操作，用相应的异常类创建对象，并使用throw关键字抛出该异常对象。

3.2 自定义异常类与异常处理

```
public void someMethod()
                        throws SomeException {
    if (someCondition()) {
        throw new SomeException("错误原因");
    }
    … … …
}
```

声明该方法可能抛出的异常

构造并抛出异常对象

调用该方法时试图捕获异常

```
… … …
try {
    someMethod();
} catch (SomeException e) {
    //异常处理代码;
}
… … …
```

定义处理异常的代码

方法是可能抛出异常的

3.3 异常类举例

```java
package shu.ces.java.chap7;

public class BankException extends Exception {
  String message;

  public BankException(int m,int n) {
     message="入账资金"+m+"是负数或支出"+n+"是正数，或者银行亏本，不符合系统要求.";
  }

  public String warnMess() {
     return message;
  }
}
```

```java
package shu.ces.java.chap7;

public class Bank {
    private int money;
    public void income(int in,int out) throws BankException {
        if(in<=0||out>=0||in+out<=0) {
            throw new BankException(in,out); //方法抛出异常，导致方法结束
        }
        int netIncome=in+out;
        System.out.printf("本次计算出的纯收入是:%d元\n",netIncome);
        money=money+netIncome;
    }
    public int getMoney() {
        return money;
    }
}
```

3.3 异常类举例

```
1  package shu.ces.java.chap7;
2
3  public class Example7_5 {
4     public static void main(String args[]) {
5        Bank bank=new Bank();
6        try{  bank.income(200,-100);
7              bank.income(300,-100);
8              bank.income(400,-100);
9              System.out.printf("银行目前有%d元\n",bank.getMoney());
10             bank.income(200, 100);
11             bank.income(99999,-100);
12       }
13       catch(BankException e) {
14             System.out.println("计算收益的过程出现如下问题:");
15             System.out.println(e.warnMess());
16       }
17       System.out.printf("银行目前有%d元\n",bank.getMoney());
18    }
19 }
```

Eclipse演示