**JAVA面向对象总结**  
本学期主要学习了JAVA面向对象，我们先来了解一下什么是面向对象.  
在程序开发的初期人们使用结构化开发语言，但是随着时间的流逝，软件的规模也越来越大，结构化语言的弊端逐渐的暴露出来，开发周期被无休止的拖延，产品的质量也越来越不尽人意，这个时候人们引入了人类最自然的思考方式，它将所有的预处理的问题抽象为对象，这就是我们所知道的面向对象，而面向对象设计的实质就是对现实世界的对象进行建模操作。  
首先我们来了解一下什么是“类”和“对象”，对同一事物的统称我们称之为类，举一个简单的例子：“在一个班里,里面有四十二名学生，尽管这些学生里面有男有女，有高矮胖瘦，这是他们的不同点，而相同点是什么，他们都有名字，学号，考试的成绩，以及和性别，我们把这些相同点抽象出来，组合成一个类，而这个类就是学生类，这是类的描述”。  
   什么是对象，在程序员的世界里一切皆对象， 对象就是事物存在的实体，我们举个简单的例子，“比如“人类”，“桌子，”，“计算机””。**只要是我们看的见摸的着的事物我们就可以称之为“对象”，而对象我们又划分为两个部分，即动态部分和静态部分，静态部分，就是不能动的部分，我们称之为“属性”，而动态部分就是能动的部分，我们称之为“方法”。  
1封装 2继承3 多态。**现在我们根据上面的理论总结一下，面向对象的特点：面向对象的三大特性。  
  
  
一 什么是“封装”{  
封装是面向对象编程的核心思想，将对象的属性和行为封装起来，这就是封装，而将对象的属性和行为封装起来的载体就是“类”， 类通常对客户隐藏其实现细节，这就是封装的思想，我们举一些简单的例子：“计算机我们都用过，我们能对计算机进行各种操作，可是我们无法看到计算机内部的结构，也不知道它是如何进行工作的，这就是“封装”。”，封装有什么好处呢，首先，封装能够保证数据的安全性，应用该类的用户不能轻易的直接操纵此数据的结构，可以完全避免外部对内部数据的影响。相对以后维护程序避免了一些不必要的麻烦。而且我们方便加入控制语句，  
封装我们主要通过get set的，get是只读，set是可写  
封装语法：  
public class Student { (定义一个类)  
        private String name; （）  
        private int age;  
        private String sex;  
        public String getName() {  
                return name;  
        }  
        public void setName(String name) {  
                this.name = name;  
        }  
        public int getAge() {  
                return age;  
        }  
        public void setAge(int age) {  
                this.age = age;  
        }  
        public String getSex() {  
                return sex;  
        }  
        public void setSex(String sex) {  
                this.sex = sex;  
        }  
}  
里面有私有的属性，我们封装一下，我们调用的时候，是通过this这个关键字来实现  
比如调用封装的属性：  
This. Name;  
This.sex;  
This.age;  
调用封装的方法也是一样的：  
This.print;  
  
}  
  
二 什么是“继承”{  
“继承”，在现实世界中，继承是指晚辈从父辈那里继承财产，也可以说是，子女拥有父母所给与他们的东西，在面向对象程序设计中，继承的含义于此类似，所不同的事，这里继承的实体是类，而不是人。  
一个班里有四十二名学生，这些学生都有共同的属性，学号，性别，考试的成绩，我们把这些属性抽象出来，组成一个学生类，这个类我们称之为“父类”，也可以说是“基类”，而每个学生都有不同的属性，比如高矮胖瘦，考试的成绩也不一样，性别也不一样，我们定义一个类，称之为“子类”，子类里面继承父类的里面的属性(子类里面也有自己的特有的属性和方法)，我们要通过extends关键字来实现：  
编写父类的语法：  
Class Pet{  
公共的属性和方法  
}  
继承的语法：  
[修饰符] class 子类名 extends 父类名{  
类体  
}  
子类访问父类成员的时候我们要通过super 这个关键字，也可以用this关键字来实现 super代表的是父类对象，  
访问父类的语法：  
Super.name访问父类的属性)  
Super.print(访问父类的一般方法)  
  
我们要注意的地方就是一个子类只能继承一个父类，不能多继承，另外不是父类里面所有的都能继承，有三点我们要注意，  
1 Private 的私有成员不能被继承，因为这是父类私有的  
2子类和父类不在同一个包中，我们也不能继承。  
3 父类的构造方法也不能被继承。  
  
继承后的初始化顺序，父类属性到父类的构造方法，再到子类属性，最后是子类的构造方法，详细的步骤，在代码中设置断点进行调试。  
  
最后继承有什么好处，首先我们能少写很多代码，不必重复，而且以后我们便于维护代码，不会很麻烦。  
}  
  
三 多态  
面向对象的核心——“多态”，随着代码量的增加和代码的维护困难，这个时候引进了多态，“多态”，是具有表现多种形态的能力的特征，专业化的说法是同一个实现接口。这就是“多态”。  
代码的多态性是面向对象的设计的重要部分，也是难以掌握的部分，在JAVA的语言中，通常使用方法重载和重写实现类的多态性，我们如何理解多态，我们举个简单的例子：“打印机”，可以分为多种打印机，比如有彩色打印机，和黑白打印机。彩色打印机，可以打印彩色的照片，而黑白打印机，只能打印黑白的照片，  
我们把打印机可以看做是一个父类，黑白打印机和彩色打印机我们可以看做是一个子类，父类打印机的方法，“打印功能”，在每个子类中都有各自不同的实现方式，比如：，对黑白打印机执行打印，打印效果是黑白的，而彩色打印机的打印效果是彩色的，很明显，这里的子类对父类的“打印方法”，进行了重写。  
这里要注意：重写是具有多态性的，是因为父类的方法在子类中被重写，虽然子类和父类的方法名相同，但是它们的功能却不一样，所以说，重写也具有多态性。  
方法重写要注意几点：  
3 私有方法不能被继承，所以也不能被重写。1在继承关系的子类中（也就是说要有继承关系。有父类和子类）。  
2 子类重写的方法，参数，返回值的类型要与父类是一些样。  
  
我们举一个简单的例子：，定义一个“人”类，具有的属性是：a 姓名，b年龄，具有的方法是“说话”。我们另外定义两个类，X类和Y类， X和Y分别继承“人”这个父类，这样X和Y都具有了“人”这个类的属性和方法， 假如X只会说汉语，而Y只会说英语， 那么是不是得在X和Y中具体去实现他们各自会说什么话，重写就是X会说英文，Y会说中文，为什么要重写，父类的描述是比较传统的，不够具体，父类描述的一些事物具有相同的地方，需要子类来描述各自不同的特点。  
  
  
  
四  “接口”，什么是“接口”，简单来说“接口”就是一种能力。  
在JAVA语言中只支持单继承，不支持多继承，也就是一个类只能有一个父类，但是在实际的应用中，又需要多继承来解决问题，这就需要到“接口”。  
举一个简单的例子，USB接口我们都知道，插上鼠标，可以让鼠标运转，插上U盘可以进行数据的传输，插上风扇，可以让风扇运转。怎样用JAVA语言来模拟呢？  
首先我们定义一个接口类   U盘类，USB风扇类。  
USB  ：  
Public interface UsbInterface{  
Void service();  
}  
  
  
USB  U盘类：  
Public class Udisk Implements UsbInterface{  
Public void service(){  
System.out printIn(“连接Usb接口，开始传输数据”);  
}  
}  
  
Usb 风扇类：  
Public class UsbFan implements UsbInterface{  
Public void service(){  
System.out printIn(“连接USB接口，风扇开始运转”);  
}  
}  
在测试类里面:  
UsbInterface uDisk =new UDisk();  
uDisk.service();  
我们发现我们在一开始定义的一个接口类，没有进行实例化，而是指向了USB风扇这个类，我们会在控制台上发现输出一句“连接USB接口，风扇开始运转”  
  
语法：  
[修饰符]interface 接口名[extends 父接口名列表]{  
[public] [static][final] 变量：  
[public]  [abstract] 方法：  
}  
注意：  
1 接口中可以定义变量，不能够定义常量。接口中的属性都会自动用“public static final”来进行修饰，也就是说，接口中的属性全部都是全局的静态常量，而且接口中的常量必须在定义的时候指定初始值。  
2 接口中所有的方法都是抽象方法，接口中方法都会自动用“public abstract ”修饰，也就是说接口中只有全局的抽象方法。  
3和抽象类一样，接口同样不能被实例化，接口中不能有构造方法。  
4 接口之间可以通过extends实现继承关系，一个接口可以继承多个接口，单接口不能继承类。  
5 一个类只能有一个直接父类，但可以通过implements 实现多个接口，类必须接口的全部方法，否则必须定义成抽象类，类在继承父类的同事又实现多个接口时，estends必须位于implementsh之前。  
  
6在类中实现接口时候，方法名，返回值类型，参数的个数以及类型必须与接口中完全一致，并且必须实现接口中所有的方法。  
在类的继承中，只能单继承，而实现接口的时候，一次则可以实现多个接口，每个接口都用逗号隔开，这时就可能出现变量或方法名冲突的情况，有变量冲突的时候，可以通过“接口名.变量”实现。

五 异常：  
异常也就是我们平时写程序遇到的BUG，如何避免异常以及出现异常我们怎么处理，在JAVA的异常中我们主要是通过五个关键字来实现，“try”，“catch”，“finally”，“throw”，“throuws”。   
下面我们看看try,catch关键字的语法：  
Try{  
代码块  
}catch(Exception e){  
代码块  
}  
  
  
public class text {  
  
        /\*\*  
         \* @param args  
         \*/  
        public static void main(String[] args) {  
                // TODO Auto-generated method stub  
                try{  
                        Scanner input =new Scanner(System.in);  
                        System.out.println("请输入除数");  
                        int num = input.nextInt();  
                        System.out.println("请输入被除数");  
                        int num1=input.nextInt();  
                        System.out.println(String.format("%d/%d=%d", num,num1,num/num1));  
                        System.out.println("感谢使用本程序");  
                }catch(Exception e){  
                        System.out.println("出现错误，被除数和除数必须是整数"+"除数不能为零");  
                        e.printStackTrace();  
                }  
                  
        }  
  
}上面这段简单的代码演示整数之间的整除，除数和被除数都只能是整数或者不能为零，但是一旦输入小数，或者除数和被除数输入了零，这就会导致程序报错或无法运行，引进try,catch,  
我们可以扑捉到这个错误，让我们发现错误的地方，进行代码的优化和修改。  
如果try语句在执行过程中碰到异常，而抛出的异常在catch快里面没有被声明，那么程序会立刻退出（）。  
  
Tyr catch finally的语法：  
  
Tyr catch finally{  
代码块  
}catch(Exception e){  
代码块  
}finally{  
代码块  
}  
public class text {  
  
        /\*\*  
         \* @param args  
         \*/  
        public static void main(String[] args) {  
                // TODO Auto-generated method stub  
                try{  
                        Scanner input =new Scanner(System.in);  
                        System.out.println("请输入除数");  
                        int num = input.nextInt();  
                        System.out.println("请输入被除数");  
                        int num1=input.nextInt();  
                        System.out.println(String.format("%d/%d=%d", num,num1,num/num1));  
                        System.out.println("感谢使用本程序");  
                }catch(Exception e){  
                        System.out.println("出现错误，被除数和除数必须是整数"+"除数不能为零");  
                        e.printStackTrace();  
                }finally{  
                        System.out.println("感谢使用本程序");  
                }  
                  
        }  
  
}  
这里我们发现最后面我们加入了一个finally语句， try catch finally，执行的流程大致分为两种情况，第一种是如果try块中所有的语句执行正常，那么finally块中的语句会被执行。第二种，如果try,catch，中的语句碰到异常，无论这种异常能否被catch扑捉到，finally中的语句也要被执行。  
在这我们要注意一点：  
即使在try块和catch块中存在return语句，finally块中的语句也会被执行，发生的异常时执行的顺序是：执行catch块或者catch中return之前的语句，执行finally中的语句，执行try块，或catch中的return语句退出。Finally块中语句唯一不执行的情况是：在异常处理代码执行System.exit(1),将退出JAVA虚拟机。  
  
多重catch块  
一段代码中出现一种异常，也可能出现多种异常，一个try catch，语句只能捕捉到一种异常，我们在try语句后面添加多个catchy语句模块，分别处理不同的异常，但排列的顺序，必须是从子类到父类，最后一个一般都是Exception类，以为按照匹配原则，如果把父类异常放到前面，后面的catch块将得不到执行的机会。  
程序运行时，系统会从头到尾分别对每个catch语句处理的异常进行检测，并执行第一个与异常匹配的catch语句。执行其中一个catch语句后，后面的catch语句也都将被忽略。  
常见的异常类型都有：  
方  法  名        说    明  
Exception         异常层次结构的根类  
ArithmeticException        算术错误情形，如以零作除数  
ArrayIndexOutOfBoundsException        数组下标越界  
NullPointerException        尝试访问 null 对象成员  
ClassNotFoundException        不能加载所需的类  
InputMismatchException        欲得到数据类型与实际输入类型不匹配  
IllegalArgumentException        方法接收到非法参数  
ClassCastException        对象强制类型转换出错  
NumberFormatException        数字格式转换异常，如把"abc"转换成数字  
  
Throw和throws  
Throws是声明异常，而throw就是抛出异常，首先try catch捕获并且处理异常，通过或throws继续声明异常，如果调用者不知道该如何处理异常，可以继续通过throws声明异常，让上一级调用者处理异常。  
  
Throw是自行抛出异常，在当前的环境无法解决参数问题时，通过throw抛出异常，把问题交给调用者去解决，简单点说就是，谁调用，谁处理异常。  
大体的异常分类有：  
6 集合框架  
我们要储存多个信息的时候，我们可以用数组来实现，但是我们会发现，数组是长度固定不变的，不能很好的使用元素数量动态变化的情况。我们定义一个30的储存空间，如果要储存大于50的信息，会造成空间不足，如果储存20的信息，会造成内存空间的浪费，我们虽然可以通过数组名.Length获取数组的长度，去无法直接获取数组中真是存在的个数，我们一旦删除数组中的元素或者增加数组中的元素时，会很麻烦。 举一个简单的例子，我们储存狗狗的信息，希望分别储存狗狗的昵称和狗狗的信息，两者具有一一对应的关系，狗狗的昵称作为狗狗信息的键存在，可以根据昵称获得狗狗的信息，这显然也无法通过数组来解决，这种情况我们引入集合。  
集合，完全完善数组的缺陷，它比数组更要灵活更加实用，大大的提高了软件的开发效率，并且不同的集合可适用于不同的场合，如果写程序并不知道程序运行时会需要多少对象，或者需要更复杂的方式储存对象，可以考虑JAVA集合来解决。  
JAVA集合框架包含的内容：  
  
  
List 接口：  
实现List接口常用类有ArrayList和LinkedList他们可以容纳所有类型的对象，包括null,允许重复，并且都保证元素的存储顺序。  
ArrayList 对数组进行了封装，实现了长度可变的数组，和数组采用相同存储方式，在内存中分配连续的空间。  
LinkedList采用链表存储方式，优点在于插入，删除元素时效率比较高，它提供了额外的addFirst(),addLast(),removelast(),等方法。可以在LinkedList的首部或者尾部进行插入或者删除操作。这些方法使得LinkedList可被用作堆栈（Stack）或者队列（queue）.  
  
  
  
7 JDBC  
为什么需要JDBC，简单点来说就是连接数据库的技术，JAVA是通过JDBC技术来实现对各种数据库的访问，换句话来说，JDBC充当了JAVA应用程序与各种不同数据库之间进行对话的媒介。  
JDBC是JAVA数据库连接（java DataBase Connectivity）技术的简称，有一组使用JAVA语言编写的类和接口组成，可以为多种关系数据库提供统一的访问。  
JDBC的工作原理。  
1 JDBC API   
2 JDBC Driver Manager  
3 JDBC 驱动   
    
JDBC访问数据库的步骤：  
1首先加载JDBC驱动  
2 与数据库建立连接  
3 发送SQL语句，并且得到返回结果  
4 处理返回结果



5771001803090012095 579036822859633082

5771001803090012386 576137399735760696

5771001803090013594 578077579902515512

5771001803090012387 577164982601818051

5771001803090012138 572131192158918326

5771001803090012359 579036822361076053

5771001803090012356 576135286143791742

5771001803090012355 575087869704693279

17088100343355274 101229944325833379

17088100343355275 101866732938832008

17088100343356107 101581152501500522

17088100343356108 101000180059871732

17088100343354295 101074194142687017

17088100343356184 101878660869628802

17088100343356185 101775831174086674

17088100343356109 101086014373572846

17088100343356110 101152207216014916

17088100343355237 101027041605702709

17088100343355238 101229364861425414

17088100343356169 101862204402635718

17088100343354928 101760654089788804