# 实验5 ：继承和多态

## 实验目的和要求

掌握父类、子类之间的关系

掌握super关键字的使用场合及方法

掌握重写的概念及与重载的区别

理解多态和动态绑定

掌握对象显示向下转换的含义及方法

掌握ArrayList的概念及使用方法

## 实验题目

### P358 11.11

改错后代码：

public class Circle {

private double radius;

public Circle(double radius)

{

this.radius = radius;

}

public double getRadius()

{

return radius;

}

public double getArea()

{

return radius \* radius \* Math.PI;

}

}

class B extends Circle

{

private double length;

B(double radius, double length)

{

super(radius);

this.length = length;

}

@Override

public double getArea() {

return super.getArea();

}

}

### P362 11.20

改错后代码：

public class Test {

public static void main(String[] args) {

Integer[] list1 = {12,24,55,1};

Double[] list2 = {12.4,24.0,55.2,1.0};

Integer[] list3 = {1,2,3};

printArray(list1);

printArray(list2);

printArray(list3);

}

public static void printArray(Object[] list)

{

for(Object o: list)

{

System.out.println(o + " ");

System.out.println();

}

}

}

### P362 11.21

代码1：

public class Test1 {

public static void main(String[] args) {

new Person().printPerson();

new Student().printPerson();

}

}

class Student extends Person

{

@Override

public String getInfo() {

return "Student";

}

}

class Person

{

public String getInfo()

{

return "Person";

}

public void printPerson()

{

System.out.println(getInfo());

}

}

结果：Person

Student

代码2：

public class Test2 {

public static void main(String[] args) {

new Person().printPerson();

new Student().printPerson();

}

}

class Student extends Person

{

private String getInfo() {

return "Student";

}

}

class Person

{

private String getInfo()

{

return "Person";

}

public void printPerson()

{

System.out.println(getInfo());

}

}

结果：Person

Person

### P366 11.26

a：true；

b：false；

c：true；

d：true；

e：false；

f：true；

g：true；

h：false；

i：可以；不可以；

j：可以；不可以；

k：不合法；

l：合法；

m：合法；

### P368 11.29

代码1：

public class Test1 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Object circle1 = new Circle();

Object circle2 = new Circle();

System.out.println(circle1.equals(circle2));

}

}

class Circle

{

double radius;

public boolean eqauls(Circle circle)

{

return this.radius == circle.radius;

}

}

结果：false

代码2：

public class Test2 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Object circle1 = new Circle();

Object circle2 = new Circle();

System.out.println(circle1.equals(circle2));

}

}

class Circle

{

double radius;

public boolean eqauls(Circle circle)

{

return this.radius == ((Circle)circle).radius;

}

}

结果：false

代码3：

**public** **class** Test3 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Circle circle1 = **new** Circle();

Circle circle2 = **new** Circle();

System.***out***.println(circle1.equals(circle2));

}

}

**class** Circle

{

**double** radius;

**public** **boolean** eqauls(Circle circle)

{

**return** **this**.radius == circle.radius;

}

}

结果：false

代码4：

**public** **class** Test4 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Circle circle1 = **new** Circle();

Circle circle2 = **new** Circle();

System.***out***.println(circle1.equals(circle2));

}

}

**class** Circle

{

**double** radius;

**public** **boolean** eqauls(Circle circle)

{

**return** **this**.radius == ((Circle)circle).radius;

}

}

结果：false

### P377 11.39

能；不能；能。

### P380 11.2

原题：（Person、Student、Employee、Faculty和Staff类）设计一个名为Person的类和它的两个名为Student和Employee的子类。Employee类又有子类：教员类Faculty和职员类Staff。每个人都有姓名、地址、电话号码和电子邮件地址。学生有班级状态（大一、大二、大三或大四）。将这些状态定义为常量。一个雇员涉及办公室、工资和受聘日期。使用编程练习题10.14中定义的MyDate类为受聘日期创建一个对象。教员有办公时间和级别。职员有职务称号。覆盖每个类中的toString方法，显示相应的类别名字个人名。

源代码：

**package** project;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Person p = **new** Person("张三","某地","110","1332@jf");

System.***out***.println(p.toString());

Student s = **new** Student("张三","某地","110","1332@jf","大一");

System.***out***.println(s.toString());

Test2 t2 = **new** Test2();

t2.setDate(561555550000L);

Employee e = **new** Employee("张三","某地","110","1332@jf","1-1",6000);

e.setYear(t2.getYear());

e.setMonth(t2.getMonth());

e.setDay(t2.getDay());

System.***out***.println(e.toString());

Faculty f = **new** Faculty("张三","某地","110","1332@jf","1-1",6000,"20:00","高级");

f.setYear(t2.getYear());

f.setMonth(t2.getMonth());

f.setDay(t2.getDay());

System.***out***.println(f.toString());

Staff st = **new** Staff("张三","某地","110","1332@jf","1-1",6000,"程序员");

st.setYear(t2.getYear());

st.setMonth(t2.getMonth());

st.setDay(t2.getDay());

System.***out***.println(st.toString());

}

}

**class** Person

{

**private** String name;

**private** String address;

**private** String pnum;

**private** String email;

**public** Person(String name, String address, String pnum, String email) {

**this**.name = name;

**this**.address = address;

**this**.pnum = pnum;

**this**.email = email;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getPnum() {

**return** pnum;

}

**public** **void** setPnum(String pnum) {

**this**.pnum = pnum;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

@Override

**public** String toString() {

**return** "name:"+name+",address:"+address+",pnum:"+pnum+",email:"+email;

}

}

**class** Student **extends** Person

{

**public** Student(String name, String address, String pnum, String email,String state) {

**super**(name, address, pnum, email);

**this**.state = state;

}

**private** String state;

@Override

**public** String toString() {

**return** **super**.toString()+",state:"+state;

}

}

**class** Employee **extends** Person

{

**public** Employee(String name, String address, String pnum, String email,String office, **double** salary) {

**super**(name, address, pnum, email);

**this**.office = office;

**this**.salary = salary;

}

**private** String office;

**private** **double** salary;

**private** **int** year;

**private** **int** month;

**private** **int** day;

**public** **void** setYear(**int** year) {

**this**.year = year;

}

**public** **void** setMonth(**int** month) {

**this**.month = month;

}

**public** **void** setDay(**int** day) {

**this**.day = day;

}

@Override

**public** String toString() {

**return** **super**.toString()+",office:"+office+",salary:"+salary+",year:"+year+",month:"+month+",day:"+day;

}

}

**class** Faculty **extends** Employee

{

**public** Faculty(String name, String address, String pnum, String email,String office, **double** salary,String time,String level) {

**super**(name, address, pnum, email, email, salary);

**this**.time = time;

**this**.level = level;

}

**private** String time;

**private** String level;

@Override

**public** String toString() {

**return** **super**.toString()+",time:"+time+",level:"+level;

}

}

**class** Staff **extends** Employee

{

**public** Staff(String name, String address, String pnum, String email,String office, **double** salary,String title) {

**super**(name, address, pnum, email, email, salary);

**this**.title = title;

}

**private** String title;

@Override

**public** String toString() {

**return** **super**.toString()+",title:"+title;

}

}

**import** java.util.GregorianCalendar;

**public** **class** Test2 {

**private** **int** year;

**private** **int** month;

**private** **int** day;

**public** Test2() {

GregorianCalendar g = **new** GregorianCalendar();

**this**.year = g.get(GregorianCalendar.***YEAR***);

**this**.month =g.get(GregorianCalendar.***MONTH***);

**this**.day =g.get(GregorianCalendar.***DAY\_OF\_MONTH***);

}

**public** Test2(**long** l) {

GregorianCalendar g = **new** GregorianCalendar();

g.setTimeInMillis(l);

**this**.year = g.get(GregorianCalendar.***YEAR***);

**this**.month =g.get(GregorianCalendar.***MONTH***);

**this**.day =g.get(GregorianCalendar.***DAY\_OF\_MONTH***);

}

**public** Test2(**int** year, **int** month, **int** day) {

**this**.year = year;

**this**.month = month;

**this**.day = day;

}

**public** **int** getYear() {

**return** year;

}

**public** **int** getMonth() {

**return** month;

}

**public** **int** getDay() {

**return** day;

}

**public** **void** setDate(**long** elapsedTime)

{

GregorianCalendar g = **new** GregorianCalendar();

g.setTimeInMillis(elapsedTime);

**this**.year = g.get(GregorianCalendar.***YEAR***);

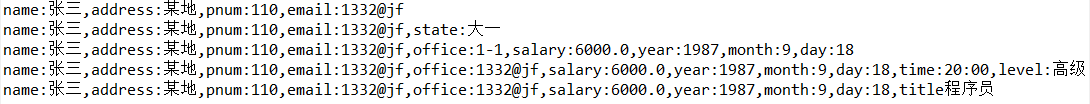
**this**.month =g.get(GregorianCalendar.***MONTH***);

**this**.day =g.get(GregorianCalendar.***DAY\_OF\_MONTH***);

}

}

结果及截图：



### P380 11.3

原题：（账户类Account的子类）在编程题练习题9.7中定义了一个Accout类来建模一个银行账户。一个账号、余额、年利率、开户日期等属性，以及存款个取款等方法。创建两个检测支票账户（checking accout）和储蓄账户（Saving accout）的子类。支票账户有一个透支限定额，但储蓄账户不能透支。

源代码：

**package** project;

**import** java.util.Date;

**public** **class** Test3 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Account a = **new** Account();

System.***out***.println(a.toString());

CheckingAccount ca = **new** CheckingAccount();

System.***out***.println(ca.toString());

SavingAccount sa = **new** SavingAccount();

System.***out***.println(sa.toString());

}

}

**class** Account

{

**private** **int** id;

**private** **double** balance;

**private** **double** annualInterestRate;

**private** Date dateCreated = **null**;

@Override

**public** String toString() {

**return** "id:"+id+",balance:"+balance+",annualInterestRate:"+annualInterestRate+",dateCreated:"+dateCreated;

}

**public** Account(){}

**public** Account(**int** id,**double** balance)

{

**this**.id = id;

**this**.balance = balance;

}

**public** **int** getId()

{

**return** id;

}

**public** **double** getBalance()

{

**return** balance;

}

**public** **double** getAnnualInterestRate()

{

**return** annualInterestRate;

}

**public** **void** setId(**int** id)

{

**this**.id = id;

}

**public** **void** setBalance(**double** balance)

{

**this**.balance = balance;

}

**public** **void** setAnnualInterestRate(**double** annualInterestRate)

{

**this**.annualInterestRate = annualInterestRate;

}

**public** Date getdateCreated()

{

**return** dateCreated;

}

**public** **double** getMonthlyInteresrRate()

{

**double** rate = annualInterestRate/12;

**return** rate;

}

**public** **void** withDraw(**double** money)

{

balance -= money;

}

**public** **void** deposit(**double** money)

{

balance += money;

}

}

**class** CheckingAccount **extends** Account

{

**private** **double** limits;

@Override

**public** String toString() {

**return** **super**.toString()+",limits:"+limits;

}

}

**class** SavingAccount **extends** Account

{

**public** String warning()

{

**return** "不能透支";

}

@Override

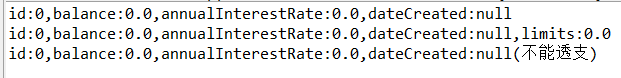
**public** String toString() {

**return** **super**.toString()+"("+warning()+")";

}

}

结果及截图：



### P380 11.13

原题：（去掉重复元素）使用下面的方法编写方法，从一个整数的数组列表中去掉重复元素：

public static void removedDuplicate(ArrayList<Integer> list)

编写测试程序，提示用户输入10个整数到列表中，显示其中不同的整数，并以一个空格分隔的方式来进行显示。这里是一个运行示例：

Enter ten integers: 34 5 3 5 6 4 33 2 2 4

The distinct Integers are 34 5 3 6 4 33 2

源代码：

import java.util.Arrays;

import java.util.Scanner;

public class Test0 {

public static void main(String[] args) {

// TODO Auto-generated method stub

int[] a = new int[10];

Scanner in = new Scanner(System.in);

System.out.println("Enter ten integers:");

for(int i=0; i<a.length; i++)

{

a[i] = in.nextInt();

}

Arrays.sort(a);

for( int i=0; i<a.length; i++)

{

System.out.print(a[i]+" ");

}

int c[] = trim(a);

System.out.println();

System.out.println("The distinct Integers are:");

for( int i=0; i<c.length; i++)

{

System.out.print(c[i]+" ");

}

}

private static int[] trim(int[] a) {

// TODO Auto-generated method stub

int i,j,k,ln;

System.out.println();

for( ln=a.length,k=i=1; i<ln; i++)

{

for(j=0; j<k; j++)

if(a[j]==a[i])

break;

if(j>=k)

a[k++]=a[i];

}

for(i=0; i<k; System.out.printf("%d ", a[i++]));

int[] b = new int[k];

for(int z=0; z<k; z++)

{

b[z] = a[z];

}

return b;

}

}

结果及截图：

