# 十二、HomeWork & Chapter08

## 一、Home: <toString用法、构造器...>

### HomeWork01

```java
package com.HomeWorkChapter08.Home01;

/*定义一个Person类(name,age,job),初始化Person对象数组
* 有3个Person对象,并按照age从大到小进行排序(用冒泡排序)*/
public class Homework01 {
    public static void main(String[] args){
        //2.初始化Person对象数组
        int len = 3;
        Person[] person = new Person[len];
        //3.有3个Person对象
        person[0] = new Person("jak",25,"快递圆儿");
        person[1] = new Person("bak",23,"打金佬");
        person[2] = new Person("cak",12,"程序员儿");

        //4.按照age从大到小进行排序 的输出
        System.out.println("原来年龄的顺序");
        for(int i = 0 ;i < len ; i++){
            System.out.println(person[i] + " ");//Person对象已经重写了toString(),因
此在这里可以直接打印Person内容
        }
        System.out.println();
        //排序后的顺序
        System.out.println("排序后的顺序");
        AgeSort ageSort = new AgeSort();
            ageSort.AgeBS(person);
            ageSort.print(person);

    }
}
```

### Person

```java
package com.HomeWorkChapter08.Home01;
//1.定义一个Person类(name,age,job)
public class Person {
    public String name;
    public int age;
    public String job;

    public Person(String name, int age,String job){
        setName(name);
        setAge(age);
        setJob(job);
    }
    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 getJob() {
        return job;
    }

    public void setJob(String job) {
        this.job = job;
    }
    public String print(){
        return getName() + "\t" + getAge() + "\t" + getJob();
    }

    @Override
    public String toString() {
        return "Person{" +
                "name='" + name + '\'' +
                ", age=" + age +
                ", job='" + job + '\'' +
                '}';
    }
}
```

## AgeSort

```
package com.HomeWorkChapter08.Home01;

public class AgeSort {//年龄排序
    Person temp = null;
    //制作一个冒泡排序功能
    public void AgeBS(Person[] person){
        for(int i = 0 ;i < person.length - 1; i++){//循环排序的组数
            for(int j = 0 ;j < person.length - 1 - i; j++){//两两元素进行对比
                if(person[j].getAge() > person[j + 1].getAge()){
                    temp = person[j];
                    person[j] = person[j + 1];
                    person[j + 1] = temp;
                }
            }
        }
    }
    //打印
    public void print(Person[] person){
        //冒泡完的顺序
        for(int i = 0 ;i < person.length; i++){
            System.out.println(person[i] + " ");//Person类用重写所以直接调用即可打印
        }
    }
}
```

## 二、Home <权限总结>

### HomeWork02

```
package com.HomeWorkChapter08.Home02;
/*题目：写出（四种访问权限修饰符） 和 (和各自的访问权限)*/
public class HomeWork02 {
    /*
        1.public ---------访问权限：同类，包，子类，不同包
        2.protected------访问权限：同类，包，子类
        3.默认--------访问权限：同类，包
        4.private---------访问权限：同类
    */
}
```

## 三、Home <继承、重写、构造器>

### HomeWork03

```
package com.HomeWorkChapter08.Home03;
/*题目：编写老师的类
*   要求：
*       1.属性：姓名(name) 年龄(age) 职称(post) 基本工资(salary)
*       2.方法：introduce(),实现一个教师的信息
*       3.编写老师类的三个子类：a.教授类  b.副教授类 c.讲师类
*         工资级别：教授:1.3w 副教授:1.2w  讲师类:1.1w
*         三个子类内：都重写父类的introduce()
*       4.定义并初始化一个老师的对象，调用业务方法，实现对象的基本信息的后台打印
*  */
public class HomeWork03 {
    public static void main(String[] args){
        Teacher[] teacher = new Teacher[3];
        teacher[0] = new Professor("张三",22,"数学");
        teacher[1] = new AssoProfessor("里尔",33,"数据结构");
        teacher[2] = new Instructor("刘武",42,"政治");
        System.out.println(teacher[0].introduce());
        System.out.println(teacher[1].introduce());
        System.out.println(teacher[2].introduce());
    }
}
```

## 父类：Teacher

```java
package com.HomeWorkChapter08.Home03;
/*题目: 编写老师的类
 *   要求:
 *       1.属性: 姓名(name) 年龄(age) 职称(post) 基本工资(salary)
 *       2.方法: introduce(),实现一个教师的信息
 *       3.编写老师类的三个子类: a.教授类  b.副教授类 c.讲师类
 *         工资级别: 教授:1.3w 副教授:1.2w  讲师类:1.1w
 *         三个子类内: 都重写父类的introduce()
 *       4.定义并初始化一个老师的对象, 调用业务方法, 实现对象的基本信息的后台打印
 *  */
public class Teacher {
    public String name;
    public int age;
    public String post;//职称

    public Teacher(String name, int age, String post) {
        setName(name);
        setAge(age);
        setPost(post);
    }
    public String introduce(){//实现一个教师的信息
        return getName() + "\t老师的年龄为: " + getAge() +
                "\t教的是: " + getPost() ;
    }

    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 getPost() {
        return post;
    }

    public void setPost(String post) {
        this.post = post;
    }
}
```

## Professor

```java
package com.HomeWorkChapter08.Home03;
/*      3.编写老师类的三个子类：a.教授类  b.副教授类 c.讲师类
 *         工资级别：教授:1.3w 副教授:1.2w  讲师类:1.1w
 *         三个子类内：都重写父类的introduce()
 *       4.定义并初始化一个老师的对象，调用业务方法，实现对象的基本信息的后台打印*/
public class Professor extends Teacher{
    private String salary = "1.3w";
    //继承类要写上父类的任意构造器
    public Professor(String name, int age, String post) {
        super(name, age, post);//引用父类的属性,作为子类的属性,并要写在开头
        setSalary(salary);
    }
    @Override
    public String introduce() {
        return super.introduce() + "\t教授工资为: " + getSalary();
    }

    public String getSalary() {
        return salary;
    }

    public void setSalary(String salary) {
        this.salary = salary;
    }
}
```

## AssoProfessor

```java
package com.HomeWorkChapter08.Home03;

public class AssoProfessor extends Teacher{
    private String salary = "1.2w";
    //子类构造器内需要super父类的属性
    public AssoProfessor(String name,int age,String post){
        super(name,age,post);
        setSalary(salary);
    }
    //重写父类introduce()
    @Override
    public String introduce() {
        return super.introduce() + "\t副教授的工资为: " + getSalary();

    }

    public String getSalary() {
        return salary;
    }

    public void setSalary(String salary) {
        this.salary = salary;
    }
}
```

### Instructor

```java
package com.HomeWorkChapter08.Home03;

public class Instructor extends Teacher{
    private String salary = "1.1w";
    public Instructor(String name, int age, String post) {
        super(name, age, post);
        setSalary(salary);
    }
    //重写父类introduce()
    @Override
    public String introduce() {
        return super.introduce() + "\t讲师的工资为" + getSalary();
    }

    public String getSalary() {
        return salary;
    }

    public void setSalary(String salary) {
        this.salary = salary;
    }
}
```

## 四、Home <继承的实例>

### HomeWork04

```java
package com.HomeWorkChapter08.Home04;
/*题目:
*   通过继承实现员工工资核算打印功能:
*   1.部门经理工资 = 10000 + 单日工资 * 天数 * 等级 (1.2)
*   2.普通员工工资 = 单日工资 * 天数 * 等级 (1.0)
*   3.员工属性: 姓名, 单日工资,工作天数
*   4.员工方法: 打印工资
*   5.普通员工 及 部门经理都是员工子类, 需要重写打印工资方法
*   6.定义并初始化普通员工的对象,调用打印工资方法输入工资,
*     定义并初始化部门经理对象,调用打印工资方法输入工资*/
public class HomeWork04 {
    public static void main(String[] args){
        Staff[] staff = new Staff[2];
        staff[0] = new OdStaff("张大壮",99.3, 360);
        staff[1] = new Manager("李小壮",199.2,360);
        staff[0].printSalary();
        staff[1].printSalary();
    }
}
```

## 父类：Staff

```
package com.HomeWorkChapter08.Home04;
/*题目:
 *    通过继承实现员工工资核算打印功能:
 *    1.部门经理工资 = 10000 + 单日工资 * 天数 * 等级 (1.2)
 *    2.普通员工工资 = 单日工资 * 天数 * 等级 (1.0)
 *    3.员工属性: 姓名, 单日工资,工作天数
 *    4.员工方法: 打印工资
 *    5.普通员工 及 部门经理都是员工子类, 需要重写打印工资方法
 *    6.定义并初始化普通员工的对象,
 *    调用打印工资方法输入工资,定义并初始化部门经理对象,
 *    调用打印工资方法输入工资*/
public class Staff {
    private String name;
    private double salaryDay;
    private int day;

    public Staff(String name, double salaryDay, int day) {
        setName(name);
        setSalaryDay(salaryDay);
        setDay(day);
    }
    //打印工资
    public void printSalary(){
        System.out.println("员工姓名: " + getName() + "\n单日工资: " +
getSalaryDay()
                + "\n工作天数" + getDay());
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getSalaryDay() {
        return salaryDay;
    }

    public void setSalaryDay(double salaryDay) {
        this.salaryDay = salaryDay;
    }

    public int getDay() {
        return day;
    }

    public void setDay(int day) {
        this.day = day;
    }
}
```

## Manager

```java
package com.HomeWorkChapter08.Home04;
/*题目:
 *   通过继承实现员工工资核算打印功能:
 *   1.部门经理工资 = 10000 + 单日工资 * 天数 * 等级 (1.2)
 *   2.普通员工工资 = 单日工资 * 天数 * 等级 (1.0)
 *   3.员工属性: 姓名, 单日工资,工作天数
 *   4.员工方法: 打印工资
 *   5.普通员工 及 部门经理都是员工子类, 需要重写打印工资方法
 *   6.定义并初始化普通员工的对象,
 *   调用打印工资方法输入工资,定义并初始化部门经理对象,
 *   调用打印工资方法输入工资*/
public class Manager extends Staff{
    private double level = 1.2;
    private double salary;

    public Manager(String name, double salaryDay, int day) {
        super(name, salaryDay, day);
        setLevel(level);
        setSalary(salary);
    }

    //重写打印方法
    public void printSalary(){
        super.printSalary();
        salary = 10000 + getSalaryDay() * getDay() * level;
        System.out.println("普通经理的工资:" + salary);
        System.out.println();
    }

    public double getLevel() {
        return level;
    }

    public void setLevel(double level) {
        this.level = level;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

## OdStaff

```java
package com.HomeWorkChapter08.Home04;
/*题目:
 *   通过继承实现员工工资核算打印功能:
 *   1.部门经理工资 = 10000 + 单日工资 * 天数 * 等级 (1.2)
 *   2.普通员工工资 = 单日工资 * 天数 * 等级 (1.0)
 *   3.员工属性: 姓名, 单日工资,工作天数
 *   4.员工方法：打印工资
 *   5.普通员工 及 部门经理都是员工子类，需要重写打印工资方法
 *   6.定义并初始化普通员工的对象，
 *   调用打印工资方法输入工资，定义并初始化部门经理对象，
 *   调用打印工资方法输入工资*/
public class OdStaff extends Staff{
    private double level = 1.2;
    private double salary;

    //构造器初始化作用:把属性都初始化
    //子类的结构体要super上父类的属性(必须放在第一行)，再把自己的属性初始化上
    public OdStaff(String name, double salaryDay, int day) {
        super(name, salaryDay, day);
        setLevel(level);
        setSalary(salary);
    }

    //重写父类打印属性
    @Override
    public void printSalary() {
        super.printSalary();//子类找不到的属性去父类找
        //普通员工工资 = 单日工资 * 天数 * 等级 (1.0)
        salary = getSalaryDay() * getDay() * level;
        System.out.println("普通员工工资: " + salary);//子类有的属性就用
        System.out.println();
    }

    public double getLevel() {
        return level;
    }

    public void setLevel(double level) {
        this.level = level;
    }

    public double getSalary(){
        return salary;
    }

    public void setSalary(double salary){
        this.salary = salary;
    }
}
```

## 五、Home <继承的实例>

### HomeWork05

```
package com.HomeWorkChapter08.Home05;
/*题目:
*   父类：员工类 子类:
*   1.工人类  2.农民类(Peasant) 3.教师类
*   4.科学家类scientist 5.服务生类waiter
*
*   (1)工人、农民服务生只有基本工资
*   (2)教师出基本工资外,还有课酬(100/天)
*   (3)科学家出基本工资外,还有年终奖
*   (4)编写一个测试类,将各种类型的员工的全年工资打印出来
* */
public class HomeWork05 {
    public static void main(String[] args){
        int len = 5;
        Staff02[] staff = new Staff02[len];
        staff[0] = new Worker("豹子",555);
        staff[1] = new Peasant("老农",1000);
        staff[2] = new Waiter("弗西斯",661);
        staff[3] = new Teacher("老大哥",5000);
        staff[4] = new Scientist("柯基",10000,5000);
        //打印全年工资
        for(int i = 0 ;i < len ; i++){
            System.out.println(staff[i].toString() + " ");
            System.out.println();
        }
    }
}
```

## 父类：Staff02

```
package com.HomeWorkChapter08.Home05;
/*题目:
 *   父类: 员工类
 *   子类:
 *   1.工人类  2.农民类(Peasant) 3.教师类
 *   4.科学家类scientist 5.服务生类waiter
 *   (1)工人、农民服务生只有基本工资
 *   (2)教师出基本工资外,还有课酬(100/天)
 *   (3)科学家出基本工资外,还有年终奖
 *   (4)编写一个测试类,将各种类型的员工的全年工资打印出来
 * */
public class Staff02 {
    private String name;
    private double baseSal;
    private int Month = 12;
    private int day = 280;
    public Staff02(String name, double baseSal) {
        this.name = name;
        this.baseSal = baseSal;
    }

    //打印全年工资


    @Override
    public String toString() {
        return "Staff02{"  + name + '\'' +
                ", 全年基本工资: " + baseSal * 12  +
                '}';
    }

    public String getName() {
        return name;
    }
```

```java
    public void setName(String name) {
        this.name = name;
    }

    public double getBaseSal() {
        return baseSal;
    }

    public void setBaseSal(double baseSal) {
        this.baseSal = baseSal;
    }

    public int getMonth() {
        return Month;
    }

    public void setMonth(int month) {
        Month = month;
    }

    public int getDay() {
        return day;
    }

    public void setDay(int day) {
        this.day = day;
    }
}
```

## Scientist

```java
package com.HomeWorkChapter08.Home05;
/*题目:
 *    父类: 员工类  子类:
 *    1.工人类   2.农民类(Peasant) 3.教师类
 *    4.科学家类scientist 5.服务生类waiter
 *
 *    (1)工人、农民服务生只有基本工资
 *    (2)教师除基本工资外,还有课酬(100/天)
 *    (3)科学家除基本工资外,还有年终奖
 *    (4)编写一个测试类,将各种类型的员工的全年工资打印出来
 * */
public class Scientist extends Staff02{
    private double bonus;//年终奖
    private double yearsal = getMonth() * getBaseSal()  + bonus;
    public Scientist(String name, double baseSal,double bonus) {
        super(name, baseSal);
        setBonus(bonus);
    }

    @Override
    public String toString() {
        return "科学家全年工资\n" + "Scientist{" + getName() + "\'"+
                ", 年终奖: " + bonus + ", 全年工资: " + yearsal +
                '}';
    }
    public double getBonus() {
        return bonus;
    }

    public void setBonus(double bonus) {
        this.bonus = bonus;
    }
}
```

## Teacher

```
package com.HomeWorkChapter08.Home05;
/*题目:
 *   父类：员工类 子类:
 *   1.工人类  2.农民类(Peasant) 3.教师类
 *   4.科学家类scientist 5.服务生类waiter
 *
 *   (1)工人、农民服务生只有基本工资
 *   (2)教师出基本工资外,还有课酬(100/天)
 *   (3)科学家出基本工资外,还有年终奖
 *   (4)编写一个测试类,将各种类型的员工的全年工资打印出来
 * */
public class Teacher extends Staff02{
    private double salary = 100;
    private double yearsal =  getMonth() * getBaseSal() + salary * getDay() ;
    public Teacher(String name, double baseSal) {
        super(name, baseSal);
    }

    @Override
    public String toString() {
        return "教师全年工资\n" + "Teacher{" + getName() + '\'' +
                ", 每天课酬: " + salary + ", 全年工资: " + yearsal +
                '}' ;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

## Peasant

```
package com.HomeWorkChapter08.Home05;
/*题目:
 *   父类：员工类 子类:
 *   1.工人类  2.农民类(Peasant) 3.教师类
 *   4.科学家类scientist 5.服务生类waiter
 *
 *   (1)工人、农民服务生只有基本工资
 *   (2)教师出基本工资外,还有课酬(100/天)
 *   (3)科学家出基本工资外,还有年终奖
 *    (4)编写一个测试类,将各种类型的员工的全年工资打印出来
 * */
public class Peasant extends Staff02{
    public Peasant(String name, double baseSal) {
        super(name, baseSal);
    }

    @Override
    public String toString() {
        return "农民的全年工资\n" + super.toString();
    }
}
```

## Waiter

```
package com.HomeWorkChapter08.Home05;

import com.HomeWorkChapter08.Home04.Staff;

/*题目:
 *    父类: 员工类 子类:
 *    1.工人类  2.农民类(Peasant) 3.教师类
 *    4.科学家类scientist 5.服务生类waiter
 *
 *    (1)工人、农民服务生只有基本工资
 *    (2)教师出基本工资外,还有课酬(100/天)
 *    (3)科学家出基本工资外,还有年终奖
 *    (4)编写一个测试类,将各种类型的员工的全年工资打印出来
 * */
public class Waiter extends Staff02 {
    public Waiter(String name, double baseSal) {
        super(name, baseSal);
    }
    @Override
    public String toString() {
        return "服务生的全年工资\n"  + super.toString();
    }
}
```

## Worker

```
package com.HomeWorkChapter08.Home05;
/*题目:
 *    父类: 员工类 子类:
 *    1.工人类  2.农民类(Peasant) 3.教师类
 *    4.科学家类scientist 5.服务生类waiter
 *
 *    (1)工人、农民服务生只有基本工资
 *    (2)教师出基本工资外,还有课酬(100/天)
 *    (3)科学家出基本工资外,还有年终奖
 *    (4)编写一个测试类,将各种类型的员工的全年工资打印出来
 * */
public class Worker extends Staff02{
    public Worker(String name, double baseSal) {
        super(name, baseSal);
    }
    @Override
    public String toString() {
        return "工人的全年工资\n" + super.toString();
    }
}
```

## 六、Home <super的用法>

### HomeWork06

```
package com.HomeWorkChapter08.Home06;
/*题目：假定Grand、Father、Son在同一个包
* 问：父类和子类中通过this和super否可以调用哪些属性和方法*/
public class HomeWork06 {
}
class Grand{
    String name ="AA";
    private int age = 100;
    public void g1(){}
}

class Father extends Grand{
    String id = "001";
    private double score;
    public void f1(){}
    /*
    super可以访问那些成员(属性)和方法？  答：可以访问父类的super.name、super.g1()
    this可以访问哪些成员？      答：this可以访问本类的this.id、this.score、this.f1()
                                   还可以访问超类的this.name、this.g1()*/

}

class Son extends Father{
    String name ="BB";
    public void g1(){}
    private void show(){}
    /*
    super可以访问那些成员(属性)和方法？  答：可以访问父类的supeer.id、super.f1(),
                                           超类的super.name、super.g1()
    this可以访问哪些成员？  答：可以访问本类的this.name、this.g1()、this.show()
                               可以访问父类的this.id、this.f1()
                               f访问不了超类的name和g1()因为本类就已经有了不用再去找上面的类
了*/

}
```

## 七、Home <多态的实例>

### HomeWork07

```
package com.HomeWorkChapter08.Home07;
/*题目：判断输出结果*/
public class HomeWork07 {
    public static void main(String[] args){
```

```java
        new Demo().test();
        /*Test
          Demo
          Rose
        * Jack*/
        new Demo("John").test();
        /*John
        * Jack */
    }
}
class Test{
    String name ="Rose";//变成了John
    Test(){
        System.out.println("Test");
    }
    Test(String name){//John
        this.name = name;//John
    }
}

class Demo extends Test{
    String name = "Jack";
    Demo(){
        super();//Test，这里的super不写也有默认的super
        System.out.println("Demo");
    }
    Demo(String s){//John
        super(s);
    }
    public void test(){
        System.out.println(super.name);//1.Rose  2.John
        System.out.println(this.name);//1.Jack   2.Jack
    }

}
```

# 八、Home

## HomeWork08

```java
package com.HomeWorkChapter08.Home08;

public class HomeWork08 {
    public static void main(String[] args){
//        CheckingAccount checkingAccount = new CheckingAccount(1000);//假设原本有1000块余额
//        checkingAccount.deposit(10);//存储10块钱，1块钱手续费 1000+(10-1)= 1009
//        checkingAccount.withdraw(9);//取款9块钱，1块手续费 1009 - (9+1) = 999
//        System.out.println("余额为:"+checkingAccount.getBalance());

        //月初，计时器自动调用earnMonthlyInterest()
        SavingAccount savingAccount = new SavingAccount(1000);
        savingAccount.deposit(100);
        savingAccount.deposit(100);
        savingAccount.deposit(100);//1300
        System.out.println("余额为: "+savingAccount.getBalance());
        savingAccount.deposit(100);//1399
        System.out.println("算手续费后的余额: "+savingAccount.getBalance());
        //余额初的重置和利息的结算
        savingAccount.earnMonthlyInterest();
        savingAccount.deposit(100);
        System.out.println("新的月份存款后的余
额: "+savingAccount.getBalance());//1399+100=1499 + (1499*0.01)
    }
}
```

## 父类：BankAccount

```
package com.HomeWorkChapter08.Home08;
/*要求:
*   1.在上面类的基础上扩展 新类CheckingAccount
*   对每次存款和取款都收取1美元的手续费
*   2.扩展前一个练习的BankAccount类，新类SavingAccount
*   每个月都会有利息产生(earnMonthlyInterest方法被调用)
*   并且有每月三次免手续费的存款或取款,
*   在earnMonthlyInterest方法中重置交易计数*/
public class BankAccount {//父类
    private double balance;//零钱
    public  BankAccount(double initialBalance){//原来零钱的余额
        this.balance = initialBalance;
    }
    //存款
    public  void deposit(double amount){
        balance += amount;
    }
    //取款
    public void withdraw(double amount){
        balance -= amount;
    }

    public double getBalance() {//get方法来查看
        return balance;
    }

    public void setBalance(double balance) {//可以调用set方法修改
        this.balance = balance;
    }
}
```

## CheckingAccount

```
package com.HomeWorkChapter08.Home08;
 /*1.在上面类的基础上扩展 新类CheckingAccount
    对每次存款和取款都收取1美元的手续费*/
public class CheckingAccount extends BankAccount{
   double charge = 1;//手续费
   public CheckingAccount(double initialBalance) {//初始余额
```

```
      super(initialBalance);
    }
   //存款---------charge：1块钱手续费
    @Override
    public void deposit(double amount) {
      super.deposit(amount - charge);
    }
   //取款
    @Override
    public void withdraw(double amount ) {
      super.withdraw(amount + charge);//这里是取钱要取的不变要多取1块钱作为手续费
    }

   }
```

## SavingAccount

```
package com.HomeWorkChapter08.Home08;
/*2.扩展前一个练习的BankAccount类，新类SavingAccount
 *   每个月都会有利息产生(earnMonthlyInterest方法被调用)
 *   并且有每月三次免手续费的存款或取款，
 *   在earnMonthlyInterest方法中重置交易计数*/
public class SavingAccount extends BankAccount{
    private int count = 3;//免手续费的次数
    private double rate = 0.01;//利息
    private int charge = 1;//手续费
    public SavingAccount(double initialBalance){
        super(initialBalance);
    }
    //存款
    @Override
    public void deposit(double amount) {
       if (count > 0){
           super.deposit(amount);
       }else{
           super.deposit(amount - charge);
       }
        count--;
    }

    //取款
    @Override
    public void withdraw(double amount) {
        if(count > 3){
            super.withdraw(amount);
        }else{
            super.withdraw(amount + charge);
        }
        count--;
    }

    public void earnMonthlyInterest(){//(1)每个月免利息的次数重置为3，(2)统计上个月的利息
        count = 3;
        super.deposit(getBalance()*rate);//利息 = 余额 * 利率
    }
}
```

## 九、Home <构造器>

### HomeWork09

```
package com.HomeWorkChapter08.Home09;
/*设计一个Point类,其x,y坐标可以通过构造器提供.
 * 提供一个子类LabeledPoint,其构造器接受一个标签值y,x坐标
 * 比如: new LabeledPoint("Black",1929,230.07),
 * 写出对应的构造器即可.*/
public class HomeWork09 {
    public static void main(String[] args){
        LabeledPoint labeledPoint = new LabeledPoint("Black",1929,230.07);
        System.out.println("构造器的名字和坐标: "+"\""+labeledPoint.getName()+"\""+
                ","+labeledPoint.getX()+","+labeledPoint.getY());
    }
}
```

### 父类Point

```
package com.HomeWorkChapter08.Home09;
/*设计一个Point类,其x,y坐标可以通过构造器提供.
* 提供一个子类LabeledPoint,其构造器接受一个标签值y,x坐标
* 比如: new LabeledPoint("Black",1929,230.07),
* 写出对应的构造器即可.*/
public class Point {
    private int x;
    private double y;


    public Point(int x, double y) {
        this.x = x;
        this.y = y;

    }

    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public double getY() {
        return y;
    }

    public void setY(double y) {
        this.y = y;
    }
}
```

### LabeledPoint

```
package com.HomeWorkChapter08.Home09;
/*提供一个子类LabeledPoint,其构造器接受一个标签值y,x坐标
 * 比如: new LabeledPoint("Black",1929,230.07)*/
public class LabeledPoint extends Point{
    private String name;
    public LabeledPoint(String name,int x, double y ) {
        super(x, y);
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

## Home10 <equals()重写>

### HomeWork10

```
package com.HomeWorkChapter08.Home10;
/*编写一个Doctor类(name,age,job,gender,sal)
 * 具有相应的setter和getter方法，五个参数的构造器，
 * 重写父类的equals(),
 * 方法：public boolean equals(Object obj),
 * 并判断测试类中创建的两个对象是否相等。(就是判断属性是否相等)
 * */
public class HomeWork10 {
    public static void main(String[] args){
        Doctor Alax = new Doctor("Alax",23,"医学博士",'男',12345);
        Doctor Alin = new Doctor("Alin", 22, "Programmer", '男', 100000);
        Doctor Alin2 = new Doctor("Alin", 22, "Programmer", '男', 100000);
        System.out.println(Alax.equals(Alin));//false
        System.out.println(Alin.equals(Alin2));//ture
    }
}
```

### Doctor

```
package com.HomeWorkChapter08.Home10;

import java.util.Objects;

/*编写一个Doctor类(name,age,job,gender,sal)
* 具有相应的setter和getter方法，五个参数的构造器，
```

```java
* 重写父类的equals(),
* 方法：public boolean equals(Object obj),
* 并判断测试类中创建的两个对象是否相等。(就是判断属性是否相等)
* */
public class Doctor {
    private String name;
    private int age;
    private String job;
    private char gender;
    private double sal;

    public Doctor(String name, int age, String job, char gender, double sal) {
        this.name = name;
        this.age = age;
        this.job = job;
        this.gender = gender;
        this.sal = sal;
    }
     public boolean  equals(Object obj){//重写父类中的equals方法
        if(this == obj)//this:本类的地址 == 传参的地址  (也就是)
            return true;
        //instanceof比较的是运行类型
        if (obj instanceof Doctor){//若obj是一个Doctor对象的话就进入
            Doctor d = (Doctor) obj;//向下转型(多态的内容)
           /* return this.name.equals(d.name) && this.age == d.age &&
                    this.job.equals(d.job) && this.gender == d.gender
                    && this.sal == d.sal;*/

        }
        return false;//若obj不是Doctor就直接返回false
     }

 /*   @Override
    public String toString() {
        return "Doctor{" +
                "name='" + name + '\'' +
                ", age=" + age +
                ", job='" + job + '\'' +
                ", gender=" + gender +
                ", sal=" + sal +
                '}';
    }*/

    /* @Override
        public boolean equals(Object o) {
            if (this == o) return true;
            if (o == null || getClass() != o.getClass()) return false;
            Doctor doctor = (Doctor) o;
            return age == doctor.age &&
                    gender == doctor.gender &&
                    Double.compare(doctor.sal, sal) == 0 &&
                    Objects.equals(name, doctor.name) &&
                    Objects.equals(job, doctor.job);
        }
    */
    @Override
    public int hashCode() {
        return Objects.hash(name, age, job, gender, sal);
    }

    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 getJob() {
        return job;
    }

    public void setJob(String job) {
        this.job = job;
    }

    public char getGender() {
        return gender;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public double getSal() {
        return sal;
    }

    public void setSal(double sal) {
        this.sal = sal;
    }
}
```

## Home11 <上下转型>

### HomeWork11

```
package com.HomeWorkChapter08.Home11;
/*现有一个Person类,里面有run(),eat(),
 * Student继承了Person类,并重写了run(),
 * 自定义了study().
 * 试写出对象向上转型 和 向下转型的代码,
 * 并写出各自都可以调用哪些方法 和 写出方法输出什么*/
public class HomeWork11 {
    public static void main(String[] args){
        //向上转型:父类的引用指向了子类对象
        Person person = new Student();
        /*person 可以调用【（Person和Students的）run()】、eat()、study()*/
        person.eat();//Person eat
        person.run();//student.run
        ((Student) person).study();//student study...

        //向下转型子类的引用强行转指向父类，父类要强制类型转换
        Student student = (Student) person;
        /*student 可以调用[student的run()]、eat()、study()*/
        student.run();//student run
        student.study();//student study...
        student.eat();//Person eat
    }
}
```

## 父类Person

```java
package com.HomeWorkChapter08.Home11;
/*现有一个Person类,里面有run(),eat(),
* Student继承了Person类,并重写了run(),
* 自定义了study().
* 试写出对象向上转型 和 向下转型的代码,
* 并写出各自都可以调用哪些方法 和 写出方法输出什么*/
public class Person {
    public void run(){
        System.out.println("Person run");
    }
    public void eat(){
        System.out.println("Person eat");
    }
}
```

## Student

```java
package com.HomeWorkChapter08.Home11;
/*现有一个Person类,里面有run(),eat(),
 * Student继承了Person类,并重写了run(),
 * 自定义了study().
 * 试写出对象向上转型 和 向下转型的代码,
 * 并写出各自都可以调用哪些方法 和 写出方法输出什么*/
public class Student extends Person{
    public void run(){
        System.out.println("student run");
    }
    public void study(){
        System.out.println("student study...");
    }
}
```

# Home12 <== 与 equals的区别>

| 名称 | 概念 | 是否可以用于基本数据类型 | 是否可以于引用类型用 |
|---|---|---|---|
| == | 比较运算符 | 可以，判断值是否相同 | 可以,判断两个对象是否相等 |
| equals | Object类的方法 java类都可以使用equals | 不可以 | 可以,默认是判断两个对象是否相等，但是子类往往重写该方法，比较的是对象的属性是否相等 |
| | | | |