Chapter 1

**Chap1\_26**

**题目要求：**输出绝对值不大于100的随机整数.当值为50时退出循环.

**程序代码：**

**import** java.util.Random;

**public** **class** chap1\_26 {

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

**int** num=0;

Random random = **new** Random();

**while**(num!=50){

num= (random.nextInt(100)) \* (random.nextInt(2)==0 ? -1 : 1);

System.out.println(num);

}

}

}

**程序流程：**

·先导入java.util包中的Random类,以此创建random对象并使用其中的方法.

·定义num来接受产生的随机整数.

·将random定义为Random对象;random.netxInt(100)用于产生0~100的随机整数;而random.nextInt(2)==0?-1:1这条语句，先是产生一个0~1的随机整数，如果等于0就乘上-;否则乘上1.以此将产生的随机整数的范围由0~100扩大到-100~100.

·while语句用于控制条件(产生的随机整数为50时退出)

·System.out.println();语句用于输出随机整数.

**程序运行结果**：(由于输出过多，仅展示部分)

-7

-91

-83

-87

8

-87

50

**Chap1\_27**

**题目要求：**输入年份，判断是否为闰年

**程序代码：**

**import** java.util.\*;

**public** **class** chap1\_27 {

//判断一年是否是闰年

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

Scanner s= **new** Scanner(System.***in***);

**while**(**true**){

System.***out***.println("请输入年份：");

**int** num=s.nextInt();

**if**((num%4==0&&num%100!=0)||num%400==0){

System.***out***.println(num+"年是闰年");

}

**else**{

System.***out***.println(num+"年不是闰年");

}

}

}

}

**程序流程：**

·导入java.util包下的所有类库

·定义输入流，用num接受读入的下一个整数(要判定是否为闰年的年份)

·if -else 条件判断是否为闰年，完成判断，输出

**程序运行结果：**

请输入年份：

2000

2000年是闰年

请输入年份：

2013

2013年不是闰年

**Chap1\_31**

**题目要求**：输入4\*4数组，按照先行后列从小到大进行排序，并输出

**程序代码：**

**import** java.util.\*;

**public** **class** chap1\_31 {

//对4\*4数组进行排序（先行后列）

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

**int** i,j;

**int** temp,counter=0;

**int** a=0;

**int** [] num =**new** **int**[16];

Scanner s=**new** Scanner(System.in);

System.out.println("请依次输入元素：");

**for**(i=0;i<=15;i++){ //读入数组

num[i]=s.nextInt();

}

**for**(i=0;i<16;i++){ //一维数组排序

**for**(j=0;j<15;j++){

**if**(num[j]<num[i]){

temp=num[i];

num[i]=num[j];

num[j]=temp;

}

}

}

**for**(i=0;i<16;i++){ //一维转换回二维并输出

**if**((i+1)%4!=0){

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

}

**else**{

System.out.println(num[i]);

}

}

}

}

**程序流程**：

·导入java.util包下的所有类库

·构造循环，依次读入4\*4数组的16个元素(先行后列)

·利用双重循环构造排序算法完成从小到大的排序

·利用for循环把一维转回二维输出

**程序运行结果：**

请依次输入元素：

1 1 2 3 4 7 8 9 5 5 11 10 1 5 2 3

11 10 9 8

7 5 5 5

4 3 3 2

2 1 1 1

Chapter2

**Chap2\_12**

**题目要求：**设计一个包含书名，作者，月销售量属性的Book类,另有两个构造方法，一个带参数，一个不带参数和两个成员方法setBook(),printBook()，前者用于设置属性，后者用于输出属性，在另一个类中编写main()方法测试.

**程序代码：**

**public** **class** chap2\_12 {

**class** Book{

String Bookname;

String Author;

**int** salevolume;

**public** Book(){

}

**public** Book(String Bookname,String Author,**int** salevolume){

}

**public** **void** setbook(String Bookname,String Author,**int** salevolume){

**this**.Bookname=Bookname;

**this**.Author=Author;

**this**.salevolume=salevolume;

}

**public** **void** printbook(){

System.***out***.println(Bookname);

System.***out***.println(Author);

System.***out***.println(salevolume);

}

}

}

-----------------------------分隔------------------------------

**import** homework.chap2\_12.Book;

**public** **class** Testchap2\_12 {

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

chap2\_12 a=**new** chap2\_12();

Book b=a.**new** Book();

b.setbook("core java","Horstmann", 200);

b.printbook();

b.Author="Joshua";

b.Bookname="effective java";

b.salevolume=100;

b.printbook();

}

}

**程序流程：**

·按照题目要求，构造好Book类

·在另一个class中，导入chap2\_12中以定义的Book类

·定义a为chap2\_12对象，并以此调用方法创建新的Book对象b

·调用b的setbook(),printbook()等函数完成测试

**程序运行结果：**

core java

Horstmann

200

effective java

Joshua

100

**Chap2\_17**

**题目要求：**设计3个类，分别是学生类Student，本科生类Undergraduate，研究生类Postgraduate，其中Student类是一个抽象类，它包含一个学生的基本信息如姓名、平均成绩等，及一个计算课程成绩等级的抽象方法。类Undergraduate和Postgraduate是其子类，它们计算课程成绩等级的方法有所不同（自己设计）。设计一个学生数组，既能放本科生对象又能放研究生对象。编写测试类测试以上功能。

**程序代码：**

**package** homework;

**public** **class** chap2\_17 {

**public** **abstract** **class** student{

**private** String name;

**private** **int** number;

**private** **float** majorlesson;

**private** **float** math;

**private** **float** english;

**public** String getName(){

**return** name;

}

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

**this**.name=name;

}

**public** **int** getnumber(){

**return** number;

}

**public** **void** setnumber(**int** number){

**this**.number=number;

}

**public** **float** getmajorlesson(){

**return** majorlesson;

}

**public** **void** setmajorlesson(**float** majorlesson){

**this**.majorlesson=majorlesson;

}

**public** **float** getMath(){

**return** math;

}

**public** **void** setMath(**float** math){

**this**.math=math;

}

**public** **float** getEnglish(){

**return** english;

}

**public** **void** setEnglish(**float** english){

**this**.english=english;

}

**public** **float** average(**float** majorlesson,**float** math,**float** english){

**float** a;

a=(majorlesson+math+english)/3;

**return** a;

}

**public** **void** scholarship(**float** average){

}

}

**public** **class** Undergraduate **extends** student{

**public** **void** scholarship(**float** average){

**if**(average>95)

System.***out***.println("该学生获得一等奖学金");

**else** **if**(average>85)

System.***out***.println("该学生获得二等奖学金");

**else** **if**(average>75)

System.***out***.println("该学生获得三等奖学金");

**else** {

System.***out***.println("该学生没有获得奖学金");

}

}

}

**public** **class** Postgraduate **extends** student{

**public** **void** scholarship(**float** average){

**if**(average>90)

System.***out***.println("该学生获得一等奖学金");

**else** **if**(average>80)

System.***out***.println("该学生获得二等奖学金");

**else** **if**(average>70)

System.***out***.println("该学生获得三等奖学金");

**else** {

System.***out***.println("该学生没有获得奖学金");

}

}

}

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

chap2\_17 c= **new** chap2\_17();

Undergraduate student1=c.**new** Undergraduate();

student1.setName("张三");

student1.setmajorlesson(81);

student1.setEnglish(88);

student1.setMath(97);

student1.setnumber(001);

System.***out***.println("本科生姓名："+student1.getName());

System.***out***.println("学生学号："+student1.getnumber());

System.***out***.println("学生专业课成绩:"+student1.getmajorlesson());

System.***out***.println("学生高数成绩："+student1.getMath());

System.***out***.println("学生大英成绩"+student1.getEnglish());

System.***out***.println("学生平均分："+student1.average(student1.getmajorlesson(),student1.getEnglish(),student1.getMath()));

System.***out***.println("学生奖学金情况：");

student1.scholarship(student1.average(student1.getmajorlesson(),student1.getEnglish(),student1.getMath()));

Postgraduate student2 = c.**new** Postgraduate();

student2.setName("李四");

student2.setmajorlesson(88);

student2.setMath(75);

student2.setEnglish(80);

student2.scholarship(91);

student2.setnumber(007);

System.***out***.println("研究生姓名："+student2.getName());

System.***out***.println("学生学号："+student2.getnumber());

System.***out***.println("学生专业课成绩:"+student2.getmajorlesson());

System.***out***.println("学生高数成绩："+student2.getMath());

System.***out***.println("学生大英成绩"+student2.getEnglish());

System.***out***.println("学生平均分："+student2.average(student2.getmajorlesson(),student2.getEnglish(),student2.getMath()));

System.***out***.println("学生奖学金情况：");

student2.scholarship(student2.average(student2.getmajorlesson(),student2.getEnglish(),student2.getMath()));

}

}

**程序流程：**

·根据题目要求设置好抽象父类student，设置好其中的各种方法和定义变量，特别的评定奖学金的办法定义为抽象方法.

·创建Undergraduate类继承student，并定义评定奖学金的方法

·创建Postgraduate类继承student，并定义评定奖学金的方法

·在main（）中，创建新的chap2\_17对象并命名为c，调用c来创建新的Undergraduate和Postgraduate对象，调用两者中定义和继承的方法，完成测试。

**程序运行结果：**

本科生姓名：张三

学生学号：1

学生专业课成绩:81.0

学生高数成绩：97.0

学生大英成绩88.0

学生平均分：88.666664

学生奖学金情况：

该学生获得二等奖学金

研究生姓名：李四

学生学号：7

学生专业课成绩:88.0

学生高数成绩：75.0

学生大英成绩80.0

学生平均分：81.0

学生奖学金情况：

该学生获得二等奖学金

**Chap2\_21**

**题目要求：**设计两个接口，每个接口包含两个方法。构造一个新接口，继承上述两个接口，并增加一个新的方法。构造一个类Test实现这个新的接口，并继承另一个类。创建类Test的实例对象，并调用类Test中全部的成员方法

**程序代码：**

**public** **class** chap2\_21 {

**interface** interA{

**void** a();

String b();

}

**interface** interB{

**void** c();

String d();

}

**interface** interC **extends** interA,interB{

**void** e();

}

}

**class** inherit{

}

**----------------------分隔----------------------**

**import** java.lang.reflect.Method;

**import** homework.chap2\_21.interC;

**public** **class** Testchap2\_21 **extends** inherit **implements** interC{

**public** **void** a(){

System.***out***.println("1");

}

**public** String b(){

**return** "咖喱gigi";

}

**public** **void** c(){

System.***out***.println("3");

}

**public** String d(){

**return** "咖喱gigi";

}

**public** **void** e(){

System.***out***.println("5");

}

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

Class<Testchap2\_21> invokertester = Testchap2\_21.**class**;

Object newInstan = invokertester.newInstance();

Method[] methods = invokertester.getDeclaredMethods();

**for**(Method m:methods) {

**if**(!"main".equals(m.getName())){

**if**(m.getReturnType()==String.**class**) {

System.***out***.println(m.invoke(newInstan,**new** Object[]{}));

}

**else** {

m.invoke(newInstan,**new** Object[]{});

}

}

}

}

}

**程序流程：**

**·定义interA和interB接口，并在每个接口中定义两个方法**

**·定义interC接口继承interA和interB，并定义一个方法**

**·定义一个新的类，导入chap2\_21中的interC**

**·将原本interC的方法，以及继承来的方法实例化，逐个测试**

**程序运行结果：**

3

咖喱gigi

咖喱gigi

1

5

**Chapter 4**

**Chap4\_6**

**题目要求：**用String，StringBuffer，StringBuilder实现10000次连接字串”abcedfg”, 显示三种类型连接用时毫秒数。

**程序代码：**

**package** homework;

**public** **class** chap4\_6 {

**public** String StringTest(){

**int** i=0;

String s1="abcdefg";

**long** l1 = System.*currentTimeMillis*();

**for**(i=0;i<10000;i++){

s1=s1+"abcdefg";

}

**long** l2 = System.*currentTimeMillis*();

System.***out***.println(l2-l1);

**return** s1;

}

**public** String StringBufferTest(){

StringBuffer s2 = **new** StringBuffer();

**long** l1 = System.*currentTimeMillis*();

**for** (**int** i = 0; i < 10000; i++) {

s2.append("abcdefg");

}

**long** l2 = System.*currentTimeMillis*();

System.***out***.println(l2-l1);

**return** s2.toString();

}

**public** **void** StringBuilderTest() {

StringBuilder s3 = **new** StringBuilder();

**long** l1 = System.*currentTimeMillis*();

**for** (**int** i=0;i<100000;i++) {

s3.append("abcdefg");

}

**long** l2 = System.*currentTimeMillis*();

System.***out***.println(l2-l1);

s3.toString();

}

}

**---------------------------分隔---------------------------**

**public** **class** chap4\_6Test {

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

chap4\_6 a = **new** chap4\_6();

a.StringTest();

a.StringBufferTest();

a.StringBuilderTest();

}

}

**程序流程：**

·创建三个方法对字符串进行10000次拼接，计算拼接开始到拼接结束的时间差来表示完成10000次拼接所需时间

·创建新的类，启动main()，定义新的chap\_4对象a,利用a调用设计好的三个方法来计算时间，评估效率

**程序输出结果：**

592

0

6

**Chap4\_8**

**题目要求：**判断命令行输入的电子邮箱是否符合格式

**程序代码：**

**package** homework;

**import** java.io.\*;

**public** **class** chap4\_8 {

**public** **static** **boolean** checkEmail(String email)

{// 验证邮箱的正则表达式

String format = "\\w{2,15}[@][a-z0-9]{2,}[.]\\p{Lower}{2,}";

//p{Alpha}:内容是必选的，和字母字符[\p{Lower}\p{Upper}]等价。如：200896@163.com不是合法的。

//w{2,15}: 2~15个[a-zA-Z\_0-9]字符；w{}内容是必选的。 如：dyh@152.com是合法的。

//[a-z0-9]{3,}：至少三个[a-z0-9]字符,[]内的是必选的；如：dyh200896@16.com是不合法的。

//[.]:'.'号时必选的； 如：dyh200896@163com是不合法的。

//p{Lower}{2,}小写字母，两个以上。如：dyh200896@163.c是不合法的。

**if** (email.matches(format))

{

**return** **true**;// 邮箱名合法，返回true

}

**else**

{

**return** **false**;// 邮箱名不合法，返回false

}

}

**public** **static** **void** main(String[] args) **throws** Exception

{

String email = "1003160184@qq.com"; // 需要进行验证的邮箱

**while**(**true**)

{

System.***out***.println("请输入要验证的邮箱:");

email = **new** BufferedReader(**new** InputStreamReader(System.***in***)).readLine();

**if** (chap4\_8.*checkEmail*(email))// 验证邮箱

{

System.***out***.println(email+"是合法的邮箱名。");

}

**else**

{

System.***out***.println(email+"不是合法的邮箱名。");

}

}

}

}

**程序流程：**

·在一个类中定义一个方法来判断输入的邮箱格式是否合法

·在另一个类中调用方法来启动main()，完成判断

**程序输出结果：**

请输入要验证的邮箱:

asjdklasjd@163.com

asjdklasjd@163.com是合法的邮箱名。

请输入要验证的邮箱:

dasjkldaj1213189@qq.com

dasjkldaj1213189@qq.com不是合法的邮箱名。

请输入要验证的邮箱:

1003160184@qq.com

1003160184@qq.com是合法的邮箱名。

**Chap4\_9**

**题目要求：**写一个计算器程序 －－扫描输入

**程序代码**：

**public** **class** chap4\_9 {

**public** **double** add(**double** x,**double** y){

**double** a;

a=x+y;

System.out.println(a);

**return** a;

}

**public** **double** subtract(**double** x,**double** y){

**double** b;

b=x-y;

System.out.println(b);

**return** b;

}

**public** **double** multiply(**double** x,**double** y){

**double** c;

c=x\*y;

System.out.println(c);

**return** c;

}

**public** **double** divide(**double** x,**double** y){

**double** d;

d=x/y;

System.out.println(d);

**return** d;

}

}

**------------------------分隔-----------------------**

**import** java.util.\*;

**public** **class** Testchap4\_9 {

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

System.out.println("欢迎使用计算器：");

**while**(**true**){

System.out.println("1.加法");

System.out.println("2.减法");

System.out.println("3.乘法");

System.out.println("4.除法");

chap4\_9 a=**new** chap4\_9();

Scanner s=**new** Scanner(System.in);

**int** b=s.nextInt();

**switch**(b){

**case** 1:{

System.out.println("请依次输入操作数(空格以分隔):");

**double** x=s.nextDouble();

**double** y=s.nextDouble();

a.add(x,y);

}

**case** 2:{

System.out.println("请依次输入操作数(空格以分隔):");

**double** x=s.nextDouble();

**double** y=s.nextDouble();

a.subtract(x,y);

}

**case** 3:{

System.out.println("请依次输入操作数(空格以分隔):");

**double** x=s.nextDouble();

**double** y=s.nextDouble();

a.multiply(x, y);

}

**case** 4:{

System.out.println("请依次输入操作数(空格以分隔):");

**double** x=s.nextDouble();

**double** y=s.nextDouble();

a.divide(x, y);

}

}

}

}

}

**程序流程：**

**·在一个类中定义加减乘除四种方法**

**·在另外一个类中定义一个变量来接受输入流，通过控制结构switch-case完成计算功能**

**程序输出结果：**

欢迎使用计算器：

1.加法

2.减法

3.乘法

4.除法

1

请依次输入操作数(空格以分隔):

4 5

9.0

(可以通过清屏API来优化)

**Chapter 7**

**Chap7\_1**

**题目要求：**将5个学生信息student添加到java.util.ArrayList，并为学生添加equals方法，equals方法判断id 相同者为同一学生，

实现功能：程序可以判断是否该学生、删除学生。列出全部学生。

学生类：student(String id,String name,Char gender,Date birtdh)

**程序代码：**

**package** homework;

**import** java.util.\*;

**public** **class** chap7\_1 {

**class** Student { // 定义学生类

String id;

String name;

String gender;

**int** birtdh;

Student(String id,String name,String gender,**int** birtdh){

**this**.id=id;

**this**.name=name;

**this**.gender=gender;

**this**.birtdh=birtdh;

}

**public** **boolean** equals(Object a){ //重写equals方法

**if** (id==**null**||a==**null**) {

**return** **false**;

}

**if** (a **instanceof** Student) {

Student b = (Student) a;

**return** id.equals(b.id);

}

**return** **false**;

}

}

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

chap7\_1 c=**new** chap7\_1();

Scanner s = **new** Scanner(System.***in***);

Student[] stu = **new** Student[4];

ArrayList<Student> stus = **new** ArrayList<Student>();

System.***out***.println("输入4个学生的id 姓名 性别 出生年份：");

**for** (**int** i = 0; i < 4; i++) {

stu[i] = c.**new** Student(s.next(),s.next(),s.next(),s.nextInt());

stus.add(stu[i]);

}

System.***out***.println("列表中还有" + stus.size() + "个学生：");

**for** (**int** i = 0; i < stus.size(); i++) { // 遍历输出所有元素中学生的姓名

System.***out***.println("第" + i + "个学生: " + stus.get(i).name);

}

System.***out***.println("输入要查找的学生的id 姓名 性别 出生年份：");

Student d = c.**new** Student(s.next(),s.nextLine(),s.next(),s.nextInt());

**if** (stus.contains(d)) {

System.***out***.println("找到" + d.name + "，下标为" + stus.indexOf(d));// 输出下标

stus.remove(d); // 从列表中移除该学生

System.***out***.println("删除" + d.name + "后,还有" + stus.size() + "个学生：");

**for** (**int** i = 0; i < stus.size(); i++) { // 遍历输出所有元素中学生的姓名

System.***out***.println("第" + i + "个学生: " + stus.get(i).name);

}

}**else**{

System.***out***.println("列表中找不到" + d.name);

}

}

**程序流程：**

·先定义学生类，并且定义学生类中包含的各种属性，重写equals方法

·在main()中去，定义chap7\_1对象并命名c

·读入学生类信息，重复调用方法，遍历输出

**程序输出结果：**

输入4个学生的id 姓名 性别 出生年份：

001 詹三 男 1998

002 叶四 男 1997

003 谢五 女 1995

004 张六 女 1997

列表中还有4个学生：

第0个学生: 詹三

第1个学生: 叶四

第2个学生: 谢五

第3个学生: 张六

输入要查找的学生的id 姓名 性别 出生年份：

002 叶四 男 1997

列表中还有3个学生：

第0个学生: 詹三

第1个学生: 谢五

第2个学生: 张六

**Chap7\_2**

**题目要求：**实现 ArrayList< student > 排序, 用户可以选择按id, 还是name, 或是gender排序

**程序代码：**

**import** java.util.\*;

**public** **class** chap7\_2 {

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

while(true){

System.out.println("请选择排序的原则:id or name or gender");

Scanner s = **new** Scanner(System.in);

String type=s.nextLine();

Comparator<Student> comparator = **new** Comparator<Student>(){

**public** **int** compare(Student s1, Student s2) {

**if**(type.equals("id")){

**return** s1.id-s2.id;

}

**else**{

**if**(type.equals("name")){

**return** s1.name.compareTo(s2.name);

}

**else**{

**return** s1.gender.compareTo(s2.gender);

}

}

}

};

Student stu1 = **new** Student (001,"张三","male");

Student stu2 = **new** Student (002,"李四","female");

Student stu3 = **new** Student (003,"王五","male");

Student stu4 = **new** Student (004,"赵六","female");

Student stu5 = **new** Student (004,"杨七","male");

ArrayList<Student> List = **new** ArrayList<Student>();

List.add(stu1);

List.add(stu2);

List.add(stu3);

List.add(stu4);

List.add(stu5);

//自动根据规则进行排序

Collections.sort(List,comparator);

display(List);

}

}

**static** **void** display(ArrayList<Student> lst){

**for**(Student s:lst)

System.out.println(s);

}

}

**class** Student{

**int** id;

String gender;

String name;

Student(**int** id,String name,String gender){

**this**.id=id;

**this**.name=name;

**this**.gender=gender;

}

**public** String toString(){

**return** id+" "+name+" "+gender;

}

}

**程序流程：**

**·**定义type来接受输入流中的对象，进行判断

·定义student类及其中的对象方法等

·判断选择排序的标准是什么，并以此进行排序

·根据排序标准排序输出，完成测试

**程序输出结果：**

请选择排序的原则:id or name or gender

id

1 张三 male

2 李四 female

3 王五 male

4 赵六 female

4 杨七 male

请选择排序的原则:id or name or gender

name

1 张三 male

2 李四 female

4 杨七 male

3 王五 male

4 赵六 female

请选择排序的原则:id or name or gender

gender

2 李四 female

4 赵六 female

1 张三 male

3 王五 male

4 杨七 male

**Chap7\_5**

**题目要求：**实现一个简单的Java关键词字典，分别使用HashMap、Treemap，存入5个以上的词条，列出所有词条。

**程序代码：**

**public** **class** HashMap<K, V> {

**private** **static** **int** DEFAULT\_CAPACITY = 16;

**private** **static** **double** A = (Math.pow(5, 0.5) - 1) / 2;

**private** **int** capacity;

**private** **int** size = 0;

**private** Node<K, V>[] buckets;

**public** HashMap() {

**this**(DEFAULT\_CAPACITY);

}

@SuppressWarnings("unchecked")

**public** HashMap(**int** capacity) {

**if** (capacity <= 0) {

**throw** **new** IllegalArgumentException(

"capacity can not be negative or zero");

}

// 保证 capacity 是2的n次方

**int** temp = 1;

**while** (temp < capacity) {

temp <<= 2;

}

**this**.capacity = temp;

buckets = **new** Node[**this**.capacity];

}

**public** **void** insert(K key, V value) {

**if** (key == **null**) {

**throw** **new** IllegalArgumentException("key can not be null");

}

**int** position = index(key);

Node<K, V> node = **new** Node<K, V>(key, value);

**if** (buckets[position] != **null**) {

node.setNext(buckets[position]);

}

buckets[position] = node;

size++;

}

**public** **void** put(K key, V value) {

**if** (key == **null**) {

**throw** **new** IllegalArgumentException("key can not be null");

}

**int** position = index(key);

Node<K, V> node = buckets[position];

**while** (node != **null**) {

**if** (node.key.equals(key)) {

node.value = value;

**return**;

}

node = node.next;

}

Node<K, V> newNode = **new** Node<K, V>(key, value);

**if** (buckets[position] != **null**) {

newNode.setNext(buckets[position]);

}

buckets[position] = newNode;

size++;

}

**public** **void** delete(K key) {

**if** (key == **null**) {

**throw** **new** IllegalArgumentException("key can not be null");

}

**int** position = index(key);

Node<K, V> node = buckets[position];

**if** (node == **null**) {

**return**;

}

**if** (node.key.equals(key)) {

buckets[position] = node.next;

size--;

}

**while** (node.next != **null**) {

**if** (node.next.key.equals(key)) {

node.next = node.next.next;

size--;

**break**;

}

node = node.next;

}

}

**public** V search(K key) {

**if** (key == **null**) {

**throw** **new** IllegalArgumentException("key can not be null");

}

**int** position = index(key);

Node<K, V> node = buckets[position];

**while** (node != **null**) {

**if** (node.key.equals(key)) {

**return** node.value;

}

node = node.next;

}

**return** **null**;

}

**public** **int** size() {

**return** size;

}

**public** **boolean** isEmpty() {

**return** size == 0;

}

@Override

**public** String toString() {

StringBuffer buffer = **new** StringBuffer();

buffer.append("{");

**for** (**int** i = 0; i < capacity; i++) {

Node<K, V> node = buckets[i];

**while** (node != **null**) {

buffer.append(node.key + ":" + node.value + ", ");

node = node.next;

}

}

**if** (buffer.length() > 1) {

buffer.delete(buffer.length() - 2, buffer.length());

}

buffer.append("}");

**return** buffer.toString();

}

**private** **int** index(K key) {

**int** hashCode = key.hashCode();

**double** temp = hashCode \* A;

**double** digit = temp - Math.floor(temp);

**return** (**int**) Math.floor(digit \* capacity);

}

**static** **class** Node<K, V> {

**private** **final** K key;

**private** V value;

**private** Node<K, V> next;

**public** Node(K key, V value) {

**this**.key = key;

**this**.value = value;

}

**public** V getValue() {

**return** value;

}

**public** **void** setValue(V value) {

**this**.value = value;

}

**public** Node<K, V> getNext() {

**return** next;

}

**public** **void** setNext(Node<K, V> next) {

**this**.next = next;

}

**public** K getKey() {

**return** key;

}

}

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

HashMap<String, String> map = **new** HashMap<String, String>();

map.put("001", "James");

map.put("002", "Antony");

map.put("003", "Bosh");

map.put("004", "Wade");

map.put("004", "WestBrook");

System.out.println(map);

System.out.println(map.size());

System.out.println(map.search("004"));

}

}

**程序流程：**

·构造hashmap

·存储词条

·输出当前词条

**程序运行结果：**

{002:Antony, 004:WestBrook, 001:James, 003:Bosh}

4

WestBrook