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：**

**package** chap1\_26;

**public** **class** abs {

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

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

**int** m = (**int**) ((Math.*random*() \* 100 + 1));

**double** f = Math.*random*();

**if** (f > 0.5) {

m = m \* (-1);

}

**if** (m == 50) {

System.***out***.println("Value is ：" + m + "，OUT！");

**break**;

} **else** {

System.***out***.println("m[" + i + "]=" + m);

}

}

}

}

**程序流程：**

·使用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.

·for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)

m[0]=-25

m[1]=23

m[2]=53

m[3]=98

m[4]=85

m[5]=70

m[6]=-67

m[7]=8

m[8]=-30

m[9]=18

m[10]=51

m[11]=-66

m[12]=32

m[13]=46

m[14]=-1

m[15]=-88

m[16]=47

m[17]=13

m[18]=-90

m[19]=-75

m[20]=19

m[21]=73

m[22]=84

m[23]=-77

m[24]=44

m[25]=-75

m[26]=-89

Value is ：50，OUT！

**Chap1\_27**

**题目要求：**输入年份，判断是否为闰年

**程序代码：**

**package** chap1\_27;

**import** java.util.Scanner;

**public** **class** leapyear {

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

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

System.***out***.println("enter year");

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

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

System.***out***.println(year+" is leap year");

}

**else**{

System.***out***.println(year+" is not leap year");

}

}

}

**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：**

enter year

2004

2004 is leap year

enter year

2006

2006 is not leap year

**Chap1\_31**

**题目要求**：输出杨辉三角

**程序代码：**

**package** chap1\_31;

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

**public** **class** yanghuitriangle {

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

**try**

{

System.***out***.print("Enter the rows of YangHui Triangle：");

BufferedReader stdin = **new** BufferedReader(**new** InputStreamReader(System.***in***));

**int** n=Integer.*parseInt*(stdin.readLine());

**int** a[][]=**new** **int**[n][];

**int** i,j;

**for**(i=0;i<n;i++)

{

a[i]=**new** **int**[i+1];

**for**(j=0;j<i+1;j++)

{

**if**(j==0||j==i)

a[i][j]=1;

**else**

a[i][j]=a[i-1][j-1]+a[i-1][j];

}

}

System.***out***.println("YangHui Triangle is：");

**for** (i=0;i<n;i++)

{

**for**(j=0;j<n-i;j++)

System.***out***.print(" ");

**for**(j=0;j<i+1;j++)

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

System.***out***.print("\n");

}

}

**catch**(IOException e)

{

System.***err***.println("IOException");

}

}

}

**程序流程**：

·导入java.util包下的所有类库

·构造循环，依次读入杨辉三角的规模

·利用二维数组进行运算

·格式化输出为等腰三角形状

**程序运行结果：**

Enter the rows of YangHui Triangle：7

YangHui Triangle is：

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 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：**

**package** chap2\_12;

**public** **class** Book{

String title;

String author;

**int** sell;

Book(){}

Book(String str1,String str2,**int** num){

title=str1;

author=str2;

sell=num;}

**public** **void** setBook(String str1,String str2,**int** num){

title=str1;

author=str2;

sell=num;}

**public** **void** printBook(){

System.***out***.println("title："+title);

System.***out***.println("author："+author);

System.***out***.println("sell："+sell);}}

**package** chap2\_12;

**public** **class** Testbook {

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

Book book1=**new** Book();

book1.setBook("C Primer Plus"," Stephen Prata",1000);

book1.printBook();

Book book2=**new** Book("C++ Primer"," Stanley B.",500);

book2.printBook();}}

**程序流程：**

·按照要求，构造Book类

·在另一个类中中，导入以定义的Book类

·调用setbook(),printbook()等函数完成测试

**程序运行结果：**

title：C Primer Plus

author： Stephen Prata

sell：1000

title：C++ Primer

author： Stanley B.

sell：500

**Chap2\_19**

**题目要求：**构建一个具有四种访问权限的成员变量和成员方法的类Test，并定义一个该类的子类，在子类中，同一包中以及不同的包中创建类Test的实例对象，看看在不同情况下试图调用所有类成员时，会得到什么样的编译信息，并根据编译结果进行分析总结

**程序代码：**

**package** chap2\_19;

**public** **class** Accessauthority {

//private的属性，只有Accessauthority自己可以访问

//子类不能继承

//其他类也不能访问

**private** **int** private\_id;

// public的属性

// 自己可以访问

// 同包子类可以继承

// 不同包子类可以继承

// 同包类可以访问

// 不同包类可以访问

**public** String public\_id;

//属性hp是default的

// 自己可以访问

// 同包子类可以继承

// 不同包子类不能继承

// 同包类可以访问

// 不同包类不能访问

**float** default\_id;

// protected饰符的属性

// 自己可以访问

// 同包子类可以继承

// 不同包子类可以继承

// 同包类可以访问

// 不同包类不能访问

**protected** **float** protected\_id;

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

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

}

}

**程序流程：**

·根据题目要求设置好Test类

·分别更改访问权限调试

**程序运行结果：**

//private的属性，只有Accessauthority自己可以访问

//子类不能继承

//其他类也不能访问

**private** **int** private\_id;

// public的属性

// 自己可以访问

// 同包子类可以继承

// 不同包子类可以继承

// 同包类可以访问

// 不同包类可以访问

**public** String public\_id;

//属性hp是default的

// 自己可以访问

// 同包子类可以继承

// 不同包子类不能继承

// 同包类可以访问

// 不同包类不能访问

**float** default\_id;

// protected饰符的属性

// 自己可以访问

// 同包子类可以继承

// 不同包子类可以继承

// 同包类可以访问

// 不同包类不能访问

**protected** **float** protected\_id;

**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：**

**package** chap2\_21;

**import** java.lang.reflect.\*;

**interface** interA{

**void** m();

String a();}

**interface** interB{

**void** n();

String b();}

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

**void** c();}

**class** inherit{}

**public** **class** Interf **extends** inherit **implements** interC{

**public** **void** m(){

System.***out***.println("print m()");}

**public** String a(){

**return** "invoke a()";}

**public** **void** n(){

System.***out***.println("print c()");}

**public** String b(){

**return** "invoke b()";}

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

System.***out***.println("print c()");}

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

Class<Interf> invokertester = Interf.**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：**

print c()

print m()

invoke a()

print c()

invoke b()

**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** chap4\_6;

**public** **class** Stringwastetime {

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

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

//StringTest();

//StringBufferTest();

*StringBuilderTest*();

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

System.***out***.println("time: " + (afterTime - beforeTime));

}

**private** **static** **void** StringTest() {

String s = "abcdefg";

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

s += "abcdefg";

}

}

**private** **static** **void** StringBufferTest() {

StringBuffer stringBuffer = **new** StringBuffer("");

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

stringBuffer = stringBuffer

.append("abcdefg");

}

}

**private** **static** **void** StringBuilderTest() {

StringBuilder stringBuilder = **new** StringBuilder("");

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

stringBuilder = stringBuilder

.append("abcdefg");

}

}

}

**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

·注释不同的方法来对结果进行比较

**程序输出结果：**

time: 452 //StringTest();

time: 2 //StringBufferTest()

time: 2 //StringBuilderTest()

**Chap4\_7**

**题目要求：**使用国际化字串，点击窗体转换 “广东工业大学”，“Guangdong University of Technology”利用中文本地资源文件和英文本地信息本地资源文件

**程序代码：**

**package chap4\_7;**

**import java.util.Locale;**

**import java.util.ResourceBundle;**

**public class Interna {**

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

**//资源包基名(包名+myproperties)**

**String basename = "myexample";**

**//设置语言环境**

**Locale cn = Locale.CHINA;//中文**

**Locale us = Locale.US;//英文**

**//根据基名和语言环境加载对应的语言资源文件**

**ResourceBundle myResourcesCN = ResourceBundle.getBundle(basename,cn);//加载myproperties\_zh.properties**

**ResourceBundle myResourcesUS = ResourceBundle.getBundle(basename,us);//加载myproperties\_en.properties**

**//加载资源文件后， 程序就可以调用ResourceBundle实例对象的 getString方法获取指定的资源信息名称所对应的值。**

**//String value = myResources.getString(“key");**

**String GDUTCN = myResourcesCN.getString("school");**

**String GDUTUS = myResourcesUS.getString("school");**

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

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

**}**

**}**

school=\u5e7f\u4e1c\u5de5\u4e1a\u5927\u5b66

school=Guangdong University of Technology

**程序流程：**

·创建properties文件来存储国际化文件

·将广东工业大学的中文译为ascII码

·调用ResourceBundle来输入结果

**程序输出结果：**

广东工业大学

Guangdong University of Technology

**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** chap4\_8;

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

**public** **class** Checkemail

{

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

{// 验证邮箱的正则表达式

String format = "\\p{Alpha}\\w{2,15}[@][a-z0-9]{3,}[.]\\p{Lower}{2,}";

**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 = "cc\*\*365@163.com"; // 需要进行验证的邮箱

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

{

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

**if** (Checkemail.*checkEmail*(email))// 验证邮箱

{

System.***out***.println(email+" is legal");

}

**else**

{

System.***out***.println(email+" is not legal");

}

}

}

}

**程序流程：**

**·设定对应的正则表达式来检测输入的格式是否正确**

**·验证并输出结果**

**程序输出结果：**

gnomeek@outlook.com

gnomeek@outlook.com is legal

g@

g@ is not legal

**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 chap7\_1;**

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

**public class Stu {**

**class Student {**

**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){**

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

**return false;**

**}**

**if (a instanceof Student) {**

**Student b = (Student) a;**

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

**}**

**return false;**

**}**

**}**

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

**Stu c=new Stu();**

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

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

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

**System.out.println("Enter four students' id name gender and birthday：");**

**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("still having" + stus.size() + "：students here");**

**for (int i = 0; i < stus.size(); i++) {**

**System.out.println("No." + i + "student: " + stus.get(i).name);}**

**System.out.println("Enter the you look for student's id name gender and birthday：");**

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

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

**System.out.println("Find" + d.name + "，Num is" + stus.indexOf(d));**

**stus.remove(d);**

**System.out.println("Delete" + d.name + "Still have" + stus.size() + "student：");**

**for (int i = 0; i < stus.size(); i++) {**

**System.out.println("No." + i + "student: " + stus.get(i).name);}}**

**else{**

**System.out.println("CANT find " + d.name); }**

**}**

**}**

**程序流程：**

·定义学生类和学生类中包含的各种属性

·重写equals方法

·读入学生类信息，重复调用方法，遍历输出

**程序输出结果：**

Enter four students' id name gender and birthday：

001 Alan male 19991111

002 Susan female 19991112

003 Jack male 19991113

003 Ponny male 19991114

still having 4：students here

No.0 student: Alan

No.1 student: Susan

No.2 student: Jack

No.3 student: Ponny

Enter the you look for student's id name gender and birthday：

001 Alan male 19991111

Still having 3： students here

002 Susan female 19991112

003 Jack male 19991113

003 Ponny male 19991114

**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：**

**package** chap7\_2;

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

**public** **class** SortStudent {

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

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

System.***out***.println("Choose the sorting ways: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,"Alan","male");

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

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

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

Student stu5 = **new** Student (004,"Ben","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并进行判断

·定义student类及其中的对象方法

·判断选择排序的标准是什么并以此进行排序

**程序输出结果：**

Choose the sorting ways:id or name or gender

id

1 Alan male

2 Susan female

3 Jack male

4 Ponny female

4 Ben male

Choose the sorting ways:id or name or gender

name

1 Alan male

4 Ben male

3 Jack male

4 Ponny female

2 Susan female

Choose the sorting ways:id or name or gender

gender

2 Susan female

4 Ponny female

1 Alan male

3 Jack male

4 Ben 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：**

**package** chap7\_5;

**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");

}

**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", "Alan");

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

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

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

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

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

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

System.***out***.println(map.search("002"));

}

}

**程序流程：**

·构造HashMap

·存储

·输出

**程序运行结果：**

{002:Susan, 004:Ben, 001:Alan, 003:Jack}

4

Ben