第 1 章 Java 概述

【1】为什么说 Java 的运行与计算机硬件平台无关？
[解答]：Java 编译器能够产生一种与计算机体系结构无关的字节指令(Byte Code)，只要安装了 Java 虚拟机，Java 就可以在相应的处理机上执行。利用 Java 虚拟机就可以把 Java 字节码程序跟具体的操作系统以及硬件平台分隔开来。

【2】Java 有什么特点？
[解答]：Java 的特点有：
(1) 简单性；
(2) 面向对象；
(3) 跨平台性；
(4) 健壮性；
(5) 安全性；
(6) 可移植性；
(7) 多线程性；
(8) 动态性。

【3】试述 Java 开发环境的建立过程。
[解答]：Java 开发环境的建立过程如下：
1．JDK 的下载与安装：从 http://java.sun.com 下载最新版的 JDK，单击下载后的文件即出现安装界面；在此界面中打击“next”按钮即可；
2．JDK 的配置：安装好 JDK 后，设安装路径为 c:\jdk1.5；还需要在环境变量中进行对应的配置。下面以 Windows 操作系统为例来进行配置。
1) 设置 JAVA_HOME 环境变量：在桌面上用鼠标右键单击“我的电脑”，依次选择“属性”、“高级”、“环境变量”选项，新建一个系统变量，名称为 JAVA_HOME，值为 c:\jdk1.5；
2) 设置 CLASSPATH 环境变量，设置方法和 JAVA_HOME 一样，其值为 c:\jdk1.5\lib\tools.jar;.;
3) 更新 PATH 环境变量的值，编辑 PATH 系统变量，在最后添加 c:\jdk1.5\bin;. 。

【4】什么是 Java API? 它提供的核心包的主要功能是什么？
[解答]：Java API 就是 Java 所提供的标准类库，它把程序设计所需要的常用的方法和接口分类封装成包。在 Java API 中主要包括核心 java 包、javax 扩展包和 org 扩展包等。核心 java 包中封装了程序设计所需要的主要应用类。

【5】如何编写和运行 Java 应用程序？
[解答]：Java 应用程序的编写和运行过程：
1．建立 Java 源文件：首先创建 Java 的源代码，即建立一个文本文档，包括有符合 Java 规范的语句。
2．编译源文件：“编译”就是将一个源代码文件翻译成计算机可以理解和处理的格式的过程。Java 源程序编译后会生成一个字节码文件，即带扩展名 class 的文件。Java 字节码文件中包含的使 Java 解释程序将要执行的指令码。

3．执行字节码文件：通过 Java 虚拟机运行字节码文件。

【6】为什么要为程序添加注释，在 Java 程序中如何为程序添加注释？
[解答]：注释是程序中的说明性文字，是程序的非执行部分。它的作用是为程序添加说明，增加程序的可读性。
Java 使用以下二种方式对程序进行注释：
1. “//”符号，它表示从“//”符号开始到此行的末尾位置都作为注释。
2. “/*. . . */”符号，它表示从“/*”开始到“*/”结束的部分都作为注释部分，可以使多行注释。

【7】Java 工具集中的 javac、java、appletviewer 各有什么作用？
[解答]：javac 的作用：它是 Java 编译器，用于将 Java 源程序编译成字节码文件；
java 的作用：它是 Java 解释器，用于解释执行 Java 字节码文件；
appletviewer 的作用：Applet 程序浏览器，用于测试和运行 Applet 程序。

【8】如何建立和运行 Java Applet 程序？
[解答]：Java Applet 程序的建立和运行过程：
1．建立 Java 源文件：首先创建 Java 的源代码，即建立一个文本文档，包括有符合 Java 规范的语句。
2．编译源文件：将 Java 源文件编译成．Class 的字节码文件，然后再将字节码文件嵌入到一个 HTML 文件中。
3．利用浏览器解释执行 html 文件。

第 2 章 Java 基础

【1】什么是数据类型？为什么要将数据划分为不同的数据类型？
[解答]：Java 是一门强类型语言。数据类型是用来区分不同的数据；由于数据在存储时所需要的容量各不相同，不同的数据就必须要分配不同大小的内存空间来存储，所有就要将数据划分成不同的数据类型。

【2】Java 中有哪些数据类型？
[解答]：Java 中有 8 种基本数据类型：字节型(byte)，短整型(short)，整型(int)，长整型(long)，字符型(char)，单精度型(float)，双精度型(double)，布尔型(boolean)。

【3】声明变量的作用是什么？
[解答]：变量时一个数据存储空间的表示，将数据指定给变量，就是将数据存储至对应的内存空间，调用变量，就是将对应的内存空间的数据取出来使用。声明变量就是申请一个内存空间。

【4】若 x=4,y=2,计算 z 值：
(1) z=x&y
(2) z=x|y
(3) z=x|y
(4) z=x>>y
(5) z= ~x
(6) z= x<< y
(7) z= x>>> y
[解答]：

```
public class Test_4 {
  public static  void main(String[] args) {
        int x = 4, y = 2, z;  //初始化 x,y,z
        z = x & y; //求 z=x&y;即 x 与 y 的值
        System.out.println("4_1:"+z);
        z = x | y; //求 z=x|y;即 x 或 y 的值
        System.out.println("4_2:"+z);
        z = x ^ y; //求 z=x^y;即 x 异或 y 的值
        System.out.println("4_3:"+z);
        z = x >> y; //求 z=x>>y;即 x 右移 y 位的值
        System.out.println("4_4:"+z);
        z = ~x; //求 z=~x;即 x 位反的值
        System.out.println("4_5:"+z);
        z = x << y; //求 z=x<<y;即 x 左移 y 位的值
        System.out.println("4_6:"+z);
        z = x >>> y; //求 z=x>>>y;即 x 无符号右移 y 位的值
        System.out.println("4_7:"+z);
```

```
    }
  }
```

【5】假设 x=10，y=20，z=30，求下列布尔表达式的值：

(1) x<10 || x<10

(2) x>y && y>x

(3) (x<y+z) && (x+10<=20)

(4) z-y==x && (y-z)==x

(5) x<10 && y>x

(6) x>y || y>x

(7) !(x<y+z) || !(x+10<=20)

(8) (!(x==y)) && (x !=y) && (x<y || y<x)

[解答]:

```
public class Test_5 {
    public static void main(String[] args) {
        int x = 10, y = 20, z = 30;
        boolean flag;
        //求 x<10||x<10 布尔值
        flag = x < 10 || x < 10;
        System.out.println("5_1:"+flag);
        //求 x>y&&y<x 布尔值
        flag = x > y && y < x;
        System.out.println("5_2:"+flag);
        //求(x<y+z)&&(x+10<=20)布尔值
        flag = (x < y + z) && (x + 10 <= 20);
        System.out.println("5_3:"+flag);
        //求 z-y==x&&(y-z)==x 布尔值
        flag = z - y == x && (y - z) == x;
        System.out.println("5_4:"+flag);
        //求 x<10&&x>10 布尔值
        flag = x < 10 && x > 10;
        System.out.println("5_5:"+flag);
        //求 x>y||y>x 布尔值
        flag = x > y || y > x;
        System.out.println("5_6:"+flag);
        //求!(x<y+z)||!(x+10<=20)布尔值
        flag = !(x < y + z) || !(x + 10 <= 20);
        System.out.println("5_7:"+flag);
        //求(!(x==y))&&(x!=y)&&(x<y||y<x)布尔值
        flag = (!(x == y)) && (x != y) && (x < y || y < x);
        System.out.println("5_8:"+flag);
    }
}
```

【6】什么是表达式？ 什么是语句？

[解答]：表达式是由运算符，操作数和方法调用按照语言的语法构造而成的符号序列。表达式可用于计算一个格式，位变量赋值以及帮助控制程序执行流程。

语句：语句组成了一个执行程序的基本单元，类似于自然语言的句子。包括表达式语句、复合语句、控制语句等。

【7】Java 有哪些数据类型？请描述其分类情况。

[解答]：Java 定义了 8 个基本数据类型：字节型，短整型，整型，长整型，字符型，浮点型，双精度型，布尔型。它们分为 4 组：

1) 整数型（byte，short，int，long）
2) 浮点数型（float，double）
3) 字符型（char）
4) 布尔型（boolean）

【8】试比较 break 和 continue 语句的区别？

[解答]：Break：强行退出循环，忽略循环体中的任何其他语句和循环的条件的条件测试。

Continue：语句用来终止本次循环。其功能是终止当前正在进行的本轮循环，即跳过后面剩余的语句，转而执行循环的第一条语句，计算和判断循环条件，决定是否进入下一轮循环。

【9】有一函数：

$$y = \begin{cases} x & (x<1) \\ 3x-2 & (1 \leqslant x<10) \\ 4x & (x \geqslant 10) \end{cases}$$

编写一程序，给定 x 值，输出 y 值。

[解答]：

```
import java.io.*;
public class Test_9 {
    public static void main(String[] args) throws NumberFormatException,IOException {
      InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        System.out.println("请输入数字 x 的值:");
        double x = Double.parseDouble(br.readLine());
        double y = 0;
        if (x < 1) {
            y = x;
        } else {
            if (x >= 1 && x < 10) {
                y = 3 * x - 2;
            } else {
                y = 4 * x;
            }
        }
```

```
            System.out.println("y=" + y);
        }
}
```

【10】说明 while 和 do……while 语句的差异。

[解答]：Do-while 与 while 语句的主要区别在于，先执行循环体中的语句再计算条件表达式，所以 do-while 语句的循环体至少执行一次。

【11】写出下列语句执行后的结果：

```
  for (k=1;k<=5;k++)
  {
  if (k>4) break;
  System.out.println(“k=”+k);
  }
```

[解答]：

```
 public class Test_11 {
      public static void main(String[] args) {
            int k;
            for (k = 1; k <= 5; k++) {
                  if (k > 4)
                        break;
                  System.out.println("k=" + k);
            }
      }
}
```

【12】编写程序，求 $\sum_{k=1}^{10} k^2$ 的值。

[解答]：

```
 public class Test_12 {
      public static void main(String[] args) {
            int k,sum=0;
            for(k=1;k<=10;k++){
                  sum=sum+k*k;
            }
            System.out.println("sum="+sum);
      }
}
```

【13】编写一程序，输入 3 个数，能按大小顺序输出。

[解答]：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
public class Test_13 {
public static void main(String[] args) throws NumberFormatException,IOException {
    InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        int a[] = new int[3];
        int k, temp;
        System.out.println("请输入 3 个数字(每输入一个数字后请换行)");
        for (int i = 0; i < 3; i++) {
            a[i] = Integer.parseInt(br.readLine());
        }
        for (int i = 0; i < a.length; i++) {
            k = i;
            for (int j = k + 1; j < a.length; j++) {
                if (a[j] < a[k]) {
                    k = j;
                }
            }
            if (k != i) {
                temp = a[i];
                a[i] = a[k];
                a[k] = temp;
            }
        }
        System.out.println("从小到大排序：");
        for (int i = 0; i < a.length; i++) {
            System.out.print(a[i] + " ");
        }
    }
}
```

【14】编写一各 Java 程序，查找 1～100 之间的素数并将运行结果输出。

[解答]:

```
public class Test_14 {
    public static void main(String[] args) {
        int m, k, i, n = 0;
        boolean flag;
        for (m = 1; m <= 100; m=m+2) {
            flag = true;
            k = (int)Math.sqrt(m);
            for (i = 2; i <= k; i++)
                if (m % i == 0) {
                    flag = false;
                    break;
                }
```

```
                if (flag) {
                    System.out.print(m + " ");
                    n = n + 1;
                }
                if (n % 10 == 0)
                    System.out.println();
            }
        }
}
```

【15】运行下面程序，并分析其执行过程：

```
Public class multiplication{
public static void main(String args[ ]){
int i,j;
for (i=1;i<10;i++){
  for (i=1;i<10;i++){
    System.out.print(i+"*"+j+"="+i*j+"");
    }
  System.out.println();
   }
   }
}
```

[解答]：

```
public class Test_15 {
    public static void main(String[] args) {
        int i, j;
        for (i = 1; i < 10; i++) {
            for (j = 1; j <= i; j++) {
                System.out.print(i + "*" + j + "=" + i * j + " ");
            }
            System.out.println();
        }
    }
}
/*
结果：
1*1=1
2*1=2 2*2=4
3*1=3 3*2=6 3*3=9
4*1=4 4*2=8 4*3=12 4*4=16
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25
6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
```

9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
乘法表
*/

【16】编写程序实现：输入一组整数，比较并输出其中的最大值和最小值，再将数组元素从小到大排序并将运行结果输出。

[解答]:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Test_16 {
    public static void main(String[] args) throws NumberFormatException,IOException {
    InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        int a[] = new int[5];
        int k, temp;
        System.out.println("请输入 5 个数字(每输入一个数字后请换行)");
        for (int i = 0; i < 5; i++) {
            a[i] = Integer.parseInt(br.readLine());
        }
        for (int i = 0; i < a.length; i++) {
            k = i;
            for (int j = k + 1; j < a.length; j++) {
                if (a[j] < a[k]) {
                    k = j;
                }
            }
            if (k != i) {
                temp = a[i];
                a[i] = a[k];
                a[k] = temp;
            }
        }
        System.out.println("最小数字 ： "+a[0]);
        System.out.println("最大数字 ： "+a[a.length-1]);
        System.out.println("从小到大排序： ");
        for (int i = 0; i < a.length; i++) {
            System.out.print(a[i] + " ");
        }
    }
}
```

【17】编写打印下列图形的程序：

(1)　　(2)　　(3)

```
#                    * * * * * * *                    $
# #                    * * * * *                    $ $ $
# # #                    * * *                    $ $ $ $ $
# # # #                    *                    $ $ $
                                                    $
```

[解答]:

```
public class Test_17_1 {
    public static void main(String[] args) {
        int i, j;
        for (i = 1; i <= 4; i++) {
            for (j = 1; j <= i; j++) {
                System.out.print("#" + " ");
            }
            System.out.println();
        }
    }
}
```

```
public class Test_17_2 {
    public static void main(String[] args) {
        int i, j, k;
        for (i = 0; i < 4; i++) {
            for (k = 0; k < i; k++) {
                System.out.print(" ");
            }
            for (j = 0; j <= 6 - 2 * i; j++)
                System.out.print("*");
            System.out.println();
        }
    }
}
```

```
public class Test_17_3 {
    public static void main(String[] args) {
        int i=0,blank,j,star,input=5;//input 是输入的行数
        String s="";
        for( i = 0;i < input; i++ )
        {
        blank = Math.abs( i - input/2);
        for( j = 1 ; j <= blank; j++ )
        s += " ";
        star = input - 2 * blank;
        for( j = 1; j <= star ; j++ )
        s += "$";
```

```
            s += "\n" ;
            }
            System.out.print( s );
        }
}
```

【18】考虑一个 2×3 的数组 a。

(1) 为 a 写一个设声明。试问，这样的声明使 a 有多少行，多少列，多少元素？

(2) 写出 a 的第 1 行的所有元素的名字。

(3) 写一条语句，置行 1 列 2 的元素为零。

(4) 写一个嵌套 for 结构，将 a 的每个元素初始化为零。

(5) 定一条语句，求第 3 列元素的和。

[解答]:

(1) 数组 a 有 2 行，3 列，元素个数为 6

(2) 第一行元素的名字为 a[0][0],a[0][1],a[0][2]

(3) 置行 1 列 2 的元素为 0,语句是 a[0][1]=0;

(4)

```
 for(int i=0;i<2;i++){
     for(in j=0;j<3;j++){
        a[i][j]=0;
        }
     }
```

(5)

```
 int sum=0;
 sum=a[0][2]+a[1][2];
```

【19】求 3×3 矩形对角元素之和。

[解答]:

```
public class Test_19 {

    public static void main(String[] args) {
        int i, j, sum = 0;
        int a[][] = new int[3][3];
        for (i = 0; i < 3; i++) {
            for (j = 0; j < 3; j++) {
                a[i][j] = i + j;
            }
        }
       //输出二维数组
        for (i = 0; i < 3; i++) {
            for (j = 0; j < 3; j++) {
                System.out.print(a[i][j] + " ");
            }
            System.out.println();
```

```
            }
        //显示对角线之和
            for (i = 0; i < 3; i++) {
                sum = sum + a[i][i];
            }
            System.out.println("对角线之和：" + sum);
    }
}
```

【20】编写一程序，查找某一字符串是否包含有 “abc”。
[解答]:

```
public class Test_20 {

    public static void main(String[] args) {
        String s = "yangen abc";
        int a = s.indexOf("abc");
        //a 位找到 abc 的下标，找不到则返回-1
        if (a != -1) {
            System.out.print("包含字符串 abc");
        } else {
            System.out.print("不包含字符串 abc");
        }
    }
}
```


【21】设一字符串中包含有大写字母的字符，也有下写字母的字符，编写一程序，将其中的大小写字母的字符分别输出。
[解答]:

```
public class Test_21 {
    public static void main(String[] args) {
        String s="sdfKJjKjjsjdfKKJkjkjSDFsdf";
        String s1="",s2="",s3;
        for(int i=0;i<s.length();i++){
        s3="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        if(s3.indexOf(s.charAt(i))!=-1){
            s1=s1+s.charAt(i);
        }else{
            s2=s2+s.charAt(i);
        }
        }
        System.out.println("大写字母 ： ");
        System.out.println(s1);
        System.out.println("小写字母 ： ");
        System.out.println(s2);
```

```
    }
}
```

【22】输出一字符串，统计其中有多少个单词。(单词之间用空格分隔)。

[解答]:

```
import java.util.*;
public class Test_22 {
    public static void main(String[] args) {
        int i = 0;
        String s = "Hello world i love java";
        StringTokenizer st = new StringTokenizer(s, " ");
        while (st.hasMoreTokens()) {
            i++;
            System.out.println(st.nextToken());
        }
        System.out.print("单词个数 :" + i);
    }
}
```

第 3 章 面向对象程序设计基础

【1】什么是 Java 程序使用的类？什么是类库？

[解答]：Java 程序的基本单位是类。对象是对事物的抽象,而类是对对象的抽象和归纳，找出事物的共性,把具有共同性质的事物归结为一类,得出一个抽象的概念——类。类是具有相同属性和方法的一组对象的集合，类是在对象之上的抽象，对象则是类的具体化，一旦建立类之后，就可用它来建立许多你需要的对象。

Java 的类库是系统提供的已实现的标准类的集合，是 Java 编程的 API(Application Program Interface)，它可以帮助开发者方便、快捷地开发 Java 程序。

【2】如何定义方法？在面向对象程序设计中方法有什么作用？

[解答]：方法的定义由两部分组成：方法声明和方法体。方法声明的基本格式如下：

返回类型　方法名（形式参数）

{

　　... //方法体内容

}

方法声明包括方法名、返回类型和形式参数，方法的小括号是方法的标志；方法体为实现方法行为的 Java 语句。

在面向对象程序设计中，方法所起的作用是完成对类和对象属性操作。

【3】简述构造方法的功能和特点。下面的程序片段是某学生为 student 类编写的构造方法，请指出其中的错误。

```
void Student(int no,String name)
{
studentNo=no;
studentName=name;
return no;
}
```

[解答]：构造方法是一个特殊的方法，主要用于初始化新创建的对象。构造方法的方法名要求与类名相同，用户不能直接调用，只能通过 new 运算符调用，而且构造方法是不返回任何数据类型，甚至也不返回 void 数据类型，即不能在构造方法前加 void。

以上的代码段出错于：①构造方法 Student()前不能加 void ②不能用 return 语句

【4】定义一个表示学生的 student 类，包括的域有学号、姓名、性别、年龄，包括的方法有获得学号、姓名、性别、年龄及修改年龄。编写 Java 程序创建 student 类的对象及测试其方法的功能。

[解答]：程序代码段如下：

```
class student{
private String id;
private String name;
private String sex;
private int age;
```

```
public String getId()
{return   id;   }
public String getName()
{ return name;   }
public String getSex()
{ return sex;   }
public int getAge()
{ return age;   }
void setAge(int age)
{ this.age = age;   }
}

public class TestStudent
{ public static void main(String args[])
{ student stu = new student("0401398", "杨小明", "男",20);
 System.out.println("student info: " + "\n 学号："+stu.getId()+"\n 姓名"+
 stu.getName()+"\n 性别："+stu.getSex()+ "\n 年龄："+stu.getAge());
 stu.setAge(19);
 System.out.println("修改后的年龄为："+stu.getAge());
}
}
```

【5】扩充、修改程序。为第 4 题的 student 类定义构造函数初始化所有的域，增加一个方法 public String printInfo()把 student 类对象的所有域信息组合形成一个字符串，并在主类中创建学生对象及测试各方法的功能。

[解答]：程序代码段如下：

```
class student{
private String id;
private String name;
private String sex;
private int age;
student(String id , String name, String sex, int age)
{ this.id = id;
this.name = name;
this.sex = sex;
this.age=age;
}
public String getId()
{return   id;   }
public String getName()
{ return name;   }
public String getSex()
{ return sex;   }
public int getAge()
```

```
{ return age;   }
void setAge(int age)
{ this.age = age;   }
public String printInfo()
{ String s= "student info: " + "\n 学号："+id+ "\n 姓名"+name+"\n 性别："+sex+ "\n 年龄"
 +age;
 System.out.println(s);
 return s;
}
}

public class TestStudent
{ public static void main(String args[])
{ student stu = new student("0401398", "杨小明", "男",20);
 stu.printInfo();
 stu.setAge(19);
 stu.printInfo();
}
}
```

【6】什么是修饰符？修饰符的种类有哪些？它们各有什么作用？

[解答]：修饰符是用于限定类对象使用的权限，从而实现类中成员的信息隐藏。访问修饰符适用于类成员，而不是方法内部的局部变量。Java 常见的访问修饰符有这 4 种：private、default、protected 和 public。

(1) public 可以被所有的类访问

(2) pravite 只能被类本身访问，其他类无法访问

(3) protected 可以被类本身、它的子类（包括同一个包中以及不同包中的子类）

(4) default 属于默认的访问状态，可以被类本身和同一个包中的类访问

【7】什么是抽象类？为什么要引入抽象类的概念？

[解答]：抽象类是用来描述人们在对问题领域进行分析、设计中得出的抽象概念，是对一系列看上去不同，但是本质上相同的具体概念的抽象。抽象类不具备实际功能，是专门设计用来让子类继承的类，把这些具有共同特征的方法抽象出来，由子类负责这些抽象方法的实现细节，而父类仅提供此方法的名称和声明、没有方法的主体,该类。

【8】什么是抽象方法？如何定义、使用抽象方法？

[解答]：用 abstract 关键字修饰的方法称为抽象方法。抽象方法定义时，需在方法名前加上关键字 abstract，抽象方法只有方法声明，没有代码实现的空方法。因此，必须通过其子类来重写父类中定义的每一个抽象方法。

【9】包的作用是什么？如何在程序中引入已定义的类？使用已定义的用户类、系统类有哪些主要方式？

[解答]：包的作用是将类和接口封装在一起，方便了类和接口的管理和调用。要引入包中已定义的类，必须用关键字 import 来导入这些类所在的包。import 语句的一般形式如下：

import 包名.类名

其中类名可以用通配符 “*”代替。

使用已定义的用户类、系统类主要有三种方式：直接调用、继承和创建已定义的用户类、系统类的对象。但无论采用哪种方式，使用已定义的用户类、系统类的前提条件是用户类、系统类应该是用户程序可见的类，为此用户程序需要用 import 语句引人它所用到的用户类、系统类或用户类、系统类所在的包。

【10】什么是继承？如何定义继承关系？

[解答]：同类事物具有共同性，在同类事物中，每个事物又具有其特殊性。运用抽象的原则舍弃对象的特殊性，抽取其共同性，则得到一个适应于一批对象的类，这便是基类（父类），而把具有特殊性的类称为派生类（子类），派生类的对象拥有其基类的全部或部分属性与方法，称作派生类对基类的继承。

派生类继承基类，必须使用关键字 extends 来声明。比如派生类 B 继承基类 A，派生类 B 定义的一般形式如下：

class B extends A

{…}

【11】什么是多态，如何实现多态？

[解答]：多态是指一个程序中同名的不同方法共存的情况。这些方法同名的原因是它们的最终功能和目的都相同，但是由于在完成同一功能时，可能遇到不同的具体情况，所以需要定义含不同的具体内容的方法，来代表多种具体实现形式。多态通常是一个消息在不同的类中用不同的方法实现的。

多态的实现是由消息的接收者确定一个消息应如何解释，而不是由消息的发送者确定，消息的发送者只需要指导另外的实例可以执行一种特定操作即可。Java 中提供两种多态机制：方法重载与方法重写。

【12】解释 this 和 super 的意义和作用。

[解答]： this 指的是引用当前对象或类名称，当一个类的实例变量名与一个方法中参数名相同时，则参数就隐藏了实例变量名，此时可通过使用关键字 this 来访问它；super 用来引用当前对象的父类，通过 super 来访问父类被子类隐藏的成员变量、成员方法以及父类的构造方法。由于子类不继承父类的构造方法，可使用 super 来调用父类的构造方法，并且 super 必须是子类构造方法中的第一条语句。

【13】什么是接口？为什么要定义接口？接口和类有什么异同？

[解答]：接口是用来调节各类之间的相互关系的一种结构，接口是抽象类的一种，只包含常量和方法的定义，而没有变量和具体方法的实现，且其方法都是抽象方法。

接口定义的格式如下：

[public] interface 接口名 [extends 父接口名列表]{

… //接口体

}

extends 子句有类声明的 extends 子句基本相同，不同的是一个接口可有多个父接口，用逗号隔开，而一个类只能有一个父类。Java 中的类只能实现单重继承，这虽然可以简化编程，但毕竟没有完全实现面向对象的功能。定义接口的主要作用，就是帮助 Java 实现类间多重继承的结构。而且接口还有以下好处：

- 接口可以被多个类所实现，这些类可以不相关，从而具有相同的行为。
- 通过接口可以了解对象的行为，而无需了解对象所对应的类的结构。

【14】将一个抽象类改写成接口，并实现这个接口。

[解答]：程序代码段如下：

```
//抽象类的应用示例
abstract  class  动物
{ public abstract void cry();  }
class 狗 extends 动物
{ public void cry()
{ System.out.println("汪汪....."); }
}
class Test
{   public static void main(String args[])
{   动物 dongwu;
    dongwu=new 狗();
    dongwu.cry();
}
}
//将抽象类改写成接口，并实现这个接口
interface 动物
{ public void cry();  }
class 狗 implements 动物
{public void cry()
{ System.out.println("汪汪....."); }
}
class Test
{   public static void main(String args[])
{   动物 dongwu;
    dongwu=new 狗();
    dongwu.cry();
}
}
```

【15】编写一个程序实现包的功能。

[解答]：包的功能实现包括两步骤：打包、引用包，其具体实现如下：

(1) //设当前运行目录的子目录 test\bag 下有 MyBag.class 类，其源程序如下：

```
package test.bag;
public class MyBag
{
public void print()
{ System.out.println("包的功能测试");}
}
```

(2) //在当前目录的 TestBag.java 中，要使用子目录 test\bag 下有 MyBag.class 类中的 print()

```
    //方法，其源程序如下：
  import test.bag.MyBag;
 public class TestBag
 { public static void main(String args[])
  { MyBag mb=new MyBag();
    mb.print();
  }
 }
```

【16】填空：

(1) 如果类 A 继承了类 B，则类 A 被称为_____类，类 B 被称为_____类。
[解答]：子　父

(2) 继承使_____成为可能，它节省了开发时间。
[解答]：代码复用

(3) 如果一个类包含一个或多个 abstract 方法，它就是一个_____类。
[解答]：抽象

(4) 一个子类一般比其超类封装的功能要_____。
[解答]：强

(5) 标记成_____的类的成员不能由该类的方法访问。
[解答]：static

(6) Java 用关键字_____指明继承关系。
[解答]：extends

(7) this 代表了_____的引用。
[解答]：当前对象

(8) super 表示的是当前对象的_____对象。
[解答]：父类

(9) 抽象类的修饰符是_____。
[解答]：abstract

(10) 接口中定义的数据成员是_____。
[解答]：常量

(11) 接口中没有什么_____方法，所有成员方法都是_____方法。
[解答]：实例　空

第 4 章 图形用户界面设计

【1】什么是图形用户界面？试列举出图形用户界面中你使用过的组件。
[解答]：图形用户界面或图形用户接口(Graphical User Interface，GUI)是指采用图形方式显示的计算机操作环境用户接口。与早期计算机使用的命令行界面相比，图形界面对于用户来说更为简便易用。（比如你用 windowsXP 和使用 DOS 操作系统的差别）。GUI 是事件驱动的，也就是说，一旦用户与 GUI 交互，GUI 组件就会生成”事件“（动作）。常见交互包括移动鼠标、单击鼠标按钮、在文字段输入、从菜单选择一个选项以及关闭一个窗口等等。
在 windwosXP 的 GUI 中，使用过窗口，菜单条，按钮等

【2】简述 Java 的事件处理机制。什么是事件源？什么是监听者？在 Java 的图形用户界面中，谁可以充当事件源？谁可以充当监听者？
[解答]：java 的事件处理机制就是，事件源允许监听器注册的事件对象，在事件发生的时候想相关的注册对象发送一个，事件对象，监听器便根据相关的信息来选择运行相关的代码。
事件源：英文名为 event source,是指事件发生的地方，也就是引起事件的组件，按钮 Button,文本组件等都可以当做事件源。比如说，你点击一个 button，那么 button 就是事件源，要想使 button 对某些事件进行响应，你就需要注册特定的监听者。（具体请看第 5 章的事件处理）
监听者：英文名为 event handler 事件处理者，又叫监听器。具体的对监听的事件类，当它监听到 event object 产生的时候，它就调用相应的方法，进行处理。
在 java 中，AWT 组件和 swing 组件都可以当做事件源；java.awt.event.*,里面各种类都可以为监听者。

【3】动作事件的事件源可以有哪些？如何响应动作事件？
[解答]：动作事件的事件源可以有：Button,JButton,MenuItem,等。
响应动作事件的过程一般为：
声明和实例化事件源，如:Button btn=new Button(“确定”);
注册监听：btn.addActionListener(this); //this 指明是在当前类实现处理
实现接口：public void actionPerformed(ActionEvent e){//具体代码};

【4】说明文本框与标签之间的区别。
[解答]：文本框（TextField）和标签(Label)都可以进行文字表达。TextField 允许用户编辑单行文本的文本组件,他可以添加相应的监听事件；而 Label 对象是一个可在容器中放置文本的组件。一个标签只显示一行只读文本。文本可由应用程序更改，但是用户不能直接对其进行编辑。

【5】编写程序包含一个标签、一个文本框和一个按钮，当用户单击按钮时，程序把文本框中的内容复制到标签中。
[解答]：Test4_5.java

```
import java.awt.*;
import java.awt.event.*;
```

```
public class Test4_5 extends Frame implements ActionListener{
  Label lb;
  TextField txtFl;
  Button btn;
  public Test4_5(){
        //界面布局和初始化
        super("文本框和标签的练习");
        setSize(260,200);
        setVisible(true);
        setLayout(new FlowLayout());
        lb=newLabel("      ");//用空格占位,以防止 label 个别字符出现问题
        txtFl=new TextField(20);
        btn=new Button("显示字符");
        add(txtFl);add(btn);
        add(lb);
        addWindowListener(new WindowAdapter(){
                    public void windowClosing(WindowEvent we){
            System.exit(0);
        }}   );

        validate();
        //增加监听
        btn.addActionListener(this);
  }

  public void actionPerformed(ActionEvent e){
      String strtmp=txtFl.getText();
      lb.setText(strtmp);
          lb.setForeground(Color.red);

  }

  public static void main(String[] args){
     new Test4_5();
  }
}
```

【6】设计一个加法计算器，如图 4.25 所示（课本 P_{124}），在文本框中输入两个整数，单击“＝”按钮时，在第三个文本框中显示这两个数据的和。

[解答]：Test4_6.java

```
import java.awt.*;
import java.awt.event.*;
class Test4_6 extends Frame implements ActionListener {
  TextField txtFl1,txtFl2,txtFl3;
```

```
    Button btn;
    public Test4_6(){
        super("简单加法运算器");
        setSize(300,200);
        setVisible(true);
        setLayout(new FlowLayout());

        txtFl1=new TextField(5);
        txtFl2=new TextField(5);
        txtFl3=new TextField(10);
        btn=new Button("=");

            add(new Label("请输入整数"));
        add(txtFl1);
        add(new Label("+"));
        add(txtFl2);
        add(btn);
        add(txtFl3);

        addWindowListener(new WindowAdapter(){
                    public void windowClosing(WindowEvent we){
            System.exit(0);
        }} );

        validate();

        btn.addActionListener(this);

    }

          public void actionPerformed(ActionEvent e){
            int                                          add=Integer.parseInt(txtFl1.getText())
+Integer.valueOf(txtFl2.getText()).intValue();//两种字符串转整形的方法
              txtFl3.setText(String.valueOf(add));
          }


    public static void main(String[] args){
       new Test4_6();
    }
  }
```

【7】说明文本框与文本区之间的区别，它们都可以引发什么事件？如何响应此事件？
[解答]：TextField 对象是允许编辑单行文本的文本组件。

TextArea 对象是显示文本的多行区域。可以将它设置为允许编辑或只读
TextFiled 和 TextArea 可以引用 KeyEvent,ActionEvent,分别用使用组件的 addKeyListener 方法注册和 addActionListener 以接收事件。

【8】设计一个计算器，其中要使用按钮、文本框、布局管理和标签等构件，能实现加、减、乘、除运算。

[解答]：Text4_8.java

```
import java.awt.*;
import java.awt.event.*;

public class Test4_8 extends Frame implements ActionListener{
        TextField txtFl1,txtFl2,txtFl3;
        Button btn1,btn2,btn3,btn4,btn5;
    public Test4_8(){
            super("简单的四则运算器");
            setSize(300,400);
            setVisible(true);
            setLayout(new GridLayout(3,1));

            txtFl1=new TextField(5);
            txtFl2=new TextField(5);
            txtFl3=new TextField(5);
            btn1=new Button("+");
            btn2=new Button("-");
            btn3=new Button("*");
            btn4=new Button("÷");

            Panel p1=new Panel();p1.add(txtFl1);p1.add(txtFl2);p1.add(new Label("="));p1.add(txtFl3);
            Panel p2=new Panel();p2.add(btn1);p2.add(btn2);p2.add(btn3);p2.add(btn4);

            add(new Label("请在下面输入运算数并运算规则进行运算:"));
            add(p1);
            add(p2);


            addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent we){
                System.exit(0);
        }}    );

        validate();
```

```
        btn1.addActionListener(this);
        btn2.addActionListener(this);
        btn3.addActionListener(this);
        btn4.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e){
        float num1=Float.valueOf(txtFl1.getText()).floatValue();
        float num2=Float.valueOf(txtFl2.getText()).floatValue();//两种字符串转整形的方
法
        float rs=0;
        if (e.getSource()==btn1){
          rs=num1+num2;
          txtFl3.setText(String.valueOf(rs));
        }
        else if (e.getSource()==btn2) {
           rs=num1-num2;
          txtFl3.setText(String.valueOf(rs));
        }
        else if (e.getSource()==btn3) {
          rs=num1*num2;
          txtFl3.setText(String.valueOf(rs));
        }
        else if (e.getSource()==btn4) {
          rs=num1/num2;
          txtFl3.setText(String.valueOf(rs));
        }
    }

  public static  void main(String[] args){
     new Test4_8();
  }
}
```

【9】创建一个窗体，窗体中有一个按钮，当单击按钮后，就会弹出一个新窗体。

[解答]：Test4_9.java

```
import java.awt.*;
import java.awt.event.*;

public class Test4_9 extends Frame implements ActionListener{
   Button btn;
   public Test4_9(){
        super("弹出窗口练习");
        setSize(300,200);
```

```
            setVisible(true);
            setLayout(new BorderLayout());

            btn=new Button("点击我会弹出窗口");
            Panel p1=new Panel();
            p1.add(new Label("      "));
            p1.add(btn);
            p1.add(new Label("      "));

            add("South",p1);
            validate();

            addWindowListener(new WindowAdapter()
            {public void windowClosing(WindowEvent e)
            {System.exit(0);}
        });

            btn.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e){
        new Test4_9();
    }


    public static void main(String[] args){
        new Test4_9();

    }
}
```

【10】编写图形界面的 Application 程序，该程序包含一个菜单，选择这个菜单的“退出”选项可以关闭 Application 的窗口并结束程序。

[解答]：Test4_10.java

```
import java.awt.*;
import java.awt.event.*;

public class Test4_10 extends Frame {
    MenuBar mbar;
    Menu file,edit,help;
    MenuItem file_open,file_new,file_exit;
    MenuItem edit_copy,edit_cut;
    MenuItem help_about;
```

```
    public Test4_10(){
        super("菜单 Application 程序");
        setSize(400,300);
        setVisible(true);

        mbar=new MenuBar();
        setMenuBar(mbar);

        file=new Menu("文件");
        edit=new Menu("编辑");
        help=new Menu("帮助");
        mbar.add(file);
        mbar.add(edit);
        mbar.add(help);

        file_new=new MenuItem("新建");
        file_open=new MenuItem("打开");
        file_exit=new MenuItem("退出");
        file.add(file_new);
        file.add(file_open);
        file.addSeparator();
        file.add(file_exit);

      edit_copy=new MenuItem("复制");
        edit_cut=new MenuItem("粘贴");
        edit.add(edit_copy);
        edit.add(edit_cut);

        help_about=new MenuItem("关于");
        help.add(help_about);


        validate();
            }
    public static  void main(String[] args){
        new Test4_10();
        }
}
```

【11】什么是容器的布局？试列举并简述 Java 中常用的几种布局策略。

[解答]：AWT 容器分为两类：外部容器和内部容器。其中，外部容器一般会独立存在，例如 Frame 类；而内部容器则会嵌套在外部容器内部使用，例如 Panel 类。容器的布局是指对添加的各个组件进行有序的、统一的对位置进行编排，使其更加美观。

（下面的例子参照网上 http://book.csdn.net/bookfiles/347/10034713585.shtml）

1．顺序布局

顺序布局（Flow Layout）是最基本的一种布局，面板的默认布局就是顺序布局。顺序布局指的是把图形元件一个接一个地放在面板上。下面是一个顺序布局的例子。

```
package sample;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
public class MyFlowLayout {
    private Frame f;
    private Button button1, button2, button3;

    public static void main (String args[]) {
        MyFlowLayout mflow = new MyFlowLayout ();
        mflow.go();
    }

    public void go() {
        f = new Frame ("FlowLayout 演示");
        f.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent evt) {
                f.setVisible(false);
                f.dispose();
                System.exit(0);
            }
        });
        //f.setLayout(new FlowLayout());
        f.setLayout(new FlowLayout(FlowLayout.LEADING, 20, 20));
        button1 = new Button("确定");
        button2 = new Button("打开");
        button3 = new Button("关闭");
        f.add(button1);
        f.add(button2);
        f.add(button3);
        f.setSize (200,200);
        f.pack();
        f.setVisible(true);
    }
}
```

程序运行结果见图 10-3。

图 10-3　顺序布局（Flow Layout）

2．边界布局

边界布局（Border Layout）包括 5 个区：北区、南区、东区、西区和中区。这 5 个区在面板上的分布规律是“上北下南，左西右东”。下面是一个边界布局的例子。

```
package sample;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
public class MyBorderLayout {
    Frame f;
    Button east, south, west, north, center;

    public static void main(String args[]) {
      MyBorderLayout mb = new MyBorderLayout();
      mb.go();
    }

    public void go() {
      f = new Frame("BorderLayout 演示");
      f.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent evt) {
          f.setVisible(false);
          f.dispose();
          System.exit(0);
        }
      });

      f.setBounds(0, 0, 300, 300);
      f.setLayout(new BorderLayout());

      north = new Button("北");
      south = new Button("南");
      east = new Button("东");
      west = new Button("西");
      center = new Button("中");

      f.add(BorderLayout.NORTH, north);
      f.add(BorderLayout.SOUTH, south);
      f.add(BorderLayout.EAST, east);
      f.add(BorderLayout.WEST, west);
      f.add(BorderLayout.CENTER, center);

      f.setVisible(true);
    }
```

```
    }
```

程序运行结果见图 10-4。

图 10-4　边界布局（Border Layout）

3．网格布局

网格布局（Grid Layout）把面板分成一个个大小相等的网格，你可以给出网格的行数和列数。下面是一个网格布局的例子。

```
package sample;
import java.awt.*;
import java.awt.event.*;
public class MyGridLayout {
    private Frame f;
    private Button[] btn;
    public static void main(String args[]) {
        MyGridLayout grid = new MyGridLayout();
        grid.go();
    }
    public void go() {
        f = new Frame("GridLayout 演示");
     f.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent evt) {
                f.setVisible(false);
                f.dispose();
                System.exit(0);
            }
        });

        f.setLayout (new GridLayout (3, 3, 10, 10));
        btn = new Button[9];
        for(int i = 0; i <=8; i++) {
            int j = i + 1;
            btn[i] = new Button("" + j);
            f.add(btn[i]);
        }

       // f.pack();
```

```
        f.setSize(100, 100);
        f.setVisible(true);
      }
    }
```

程序运行结果见图 10-5。

图 10-5　网格布局（Grid Layout）

4．卡片布局

卡片布局（Card Layout）把每个组件看作一张卡片，好像一副扑克牌，它们叠在一起，每次只有最外面的一个组件可以被看到。

```
package sample;
import java.awt.*;
import java.awt.event.*;
public class MyCardLayout {
  public static void main(String args[]) {
    new MyCardLayout().go();
  }

  public void go() {
    final Frame f = new Frame("CardLayout 演示");
    f.addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent evt) {
        f.setVisible(false);
        f.dispose();
        System.exit(0);
      }
    });

    f.setSize(300, 100);
    f.setLayout(new CardLayout());

    final Frame f1 = f;
    for(int i = 1; i <= 5; ++i) {
      Button b = new Button("Button " + i);
      b.setSize(100, 25);
      b.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ae) {
```

```
                CardLayout cl = (CardLayout)f1.getLayout();
                cl.next(f1);
              }
            } );
            f.add(b, "button" + i);
          }
          f.setVisible(true);
        }
    }
```

程序运行结果见图 10-6。

图 10-6 卡片布局（Card Layout）

单击按钮 Button1 后，显示下一个按钮 Button2，依此类推。

5．网格包布局

网格包 （GridBag）布局是基于网格布局之上的一种改进。和基本的网格布局不同的是，一个组件可以跨越一个或多个网格，这样一来增加了布局的灵活性。为了处 理网格的跨越性，我们可以使用 GridBagConstraints 类。有兴趣的读者可以参考 Java API 来了解它。

```
package sample;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
 public class MyGridBagLayout extends Panel {
    protected void makebutton(String name,
                              GridBagLayout gridbag,
                              GridBagConstraints c) {
       Button button = new Button(name);
       gridbag.setConstraints(button, c);
       add(button);
    }
    public void go() {
       GridBagLayout gridbag = new GridBagLayout();
       GridBagConstraints c = new GridBagConstraints();
       setFont(new Font("Helvetica", Font.PLAIN, 14));
       setLayout(gridbag);
       c.fill = GridBagConstraints.BOTH;
       c.weightx = 1.0;
       makebutton("Button001", gridbag, c);
```

```
        makebutton("Button2", gridbag, c);
        makebutton("Button3", gridbag, c);
          c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button4", gridbag, c);
        c.weightx = 0.0;          //reset to the default
        makebutton("Button5", gridbag, c); //another row
       c.gridwidth = 2; //GridBagConstraints.RELATIVE; //next-to-last in row
        makebutton("Button6", gridbag, c);
       c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button007", gridbag, c);
       c.gridwidth = 1;              //reset to the default
       c.gridheight = 2;
        c.weighty = 1.0;
        makebutton("Button8", gridbag, c);
        c.weighty = 1.0;          //reset to the default
       c.gridwidth = GridBagConstraints.REMAINDER; //end row
       c.gridheight = 1;          //reset to the default
        makebutton("Button9", gridbag, c);
        makebutton("Button10", gridbag, c);
        setSize(300, 100);
    }
    public static void main(String args[]) {
      final Frame f = new Frame("GridBagLayout 演示");
        f.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent evt) {
              f.setVisible(false);
              f.dispose();
              System.exit(0);
            }
        });
      MyGridBagLayout gb = new MyGridBagLayout();
      gb.go();
      f.add("Center", gb);
      f.pack();
      f.setVisible(true);
    }
}
```

程序运行结果见图 10-7。

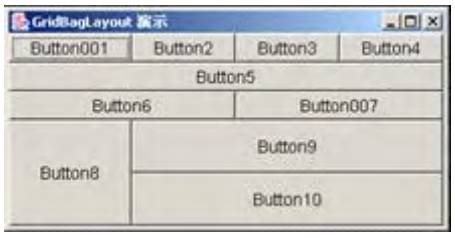


图 10-7　网格包（GridBag）布局

【12】根据本章所学的内容编程：设计一个模拟的文字编辑器，并用菜单实现退出的功能。

[解答]：Test4_12.java

```
import java.awt.*;
import java.awt.event.*;

public class Test4_12 extends Frame implements ActionListener{
   MenuBar mbar;
   Menu file,edit,help;
   MenuItem file_open,file_new,file_save,file_exit;
   MenuItem edit_copy,edit_cut,edit_pase,edit_del;
   MenuItem help_about;

   Button btn_copy,btn_pase,btn_cut,btn_del;

   TextArea txtAr;
   boolean ifdo;

   StringBuffer strtmp;

   public Test4_12(){
      super("菜单 Application 程序");
      setSize(400,300);
      setVisible(true);
      setLayout(new BorderLayout());

      mbar=new MenuBar();
      setMenuBar(mbar);

      strtmp=new StringBuffer();

      file=new Menu("文件");
      edit=new Menu("编辑");
      help=new Menu("帮助");
      mbar.add(file);
      mbar.add(edit);
      mbar.add(help);

      file_new=new MenuItem("新建");
      file_open=new MenuItem("打开");
      file_open.setEnabled(false);
      file_save=new MenuItem("保存");
      file_save.setEnabled(false);
```

```
file_exit=new MenuItem("退出");
file.add(file_new);
file.add(file_open);
file.add(file_save);
file.addSeparator();
file.add(file_exit);

edit_copy=new MenuItem("复制",new MenuShortcut(KeyEvent.VK_C));
edit_pase=new MenuItem("粘贴",new MenuShortcut(KeyEvent.VK_V));
edit_cut=new MenuItem("剪切",new MenuShortcut(KeyEvent.VK_X));
edit_del=new MenuItem("删除",new MenuShortcut(KeyEvent.VK_DELETE));
edit.add(edit_copy);
edit.add(edit_pase);
edit.add(edit_cut);
edit.add(edit_del);

help_about=new MenuItem("关于");
help.add(help_about);

txtAr=new TextArea(8,10);
Panel p1=new Panel();
btn_copy=new Button("复制");
btn_cut=new Button("剪切");
btn_pase=new Button("粘贴");
btn_del=new Button("删除");
p1.setLayout(new FlowLayout(FlowLayout.LEFT));
p1.add(btn_copy);p1.add(btn_cut);p1.add(btn_pase);p1.add(btn_del);
add("North",p1);
add("Center",txtAr);
validate();

file_new.addActionListener(this);
file_open.addActionListener(this);
file_save.addActionListener(this);
file_exit.addActionListener(this);
edit_copy.addActionListener(this);
edit_pase.addActionListener(this);
edit_cut.addActionListener(this);
edit_del.addActionListener(this);
help_about.addActionListener(this);
btn_copy.addActionListener(this);
btn_cut.addActionListener(this);
btn_pase.addActionListener(this);
btn_del.addActionListener(this);
```

```
        addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent we){
                System.exit(0);
            }}    );
    }

    public void actionPerformed(ActionEvent e){
     if (e.getSource()==file_new){
        txtAr.setText(null);
     }
     else if (e.getSource()==file_open){
     }
     else if (e.getSource()==file_save){
     }
     else if (e.getSource()==file_exit){
       System.exit(0);     //退出
     }
     else if (e.getSource()==edit_copy||e.getSource()==btn_copy){   //复制的实现
       strtmp.delete(0,strtmp.length());
       strtmp.append(txtAr.getSelectedText());
     }
     else if (e.getSource()==edit_pase||e.getSource()==btn_pase){   //粘贴的实现
       txtAr.insert(strtmp.toString(),txtAr.getSelectionEnd());
     }
     else if (e.getSource()==edit_cut||e.getSource()==btn_cut){     //剪切的实现
         strtmp.delete(0,strtmp.length());
         strtmp.append(txtAr.getSelectedText());
         String strtmp1=new String(txtAr.getText().substring(0,txtAr.getSelectionStart()));
         String                                                          strtmp2=new
String(txtAr.getText().substring(txtAr.getSelectionEnd(),txtAr.getText().length()));    //返回一个
新的 String，它包含此序列当前所包含的字符子序列。
         txtAr.setText(strtmp1+strtmp2);
     }
     else if(e.getSource()==edit_del||e.getSource()==btn_del){
         String strtmp1=new String(txtAr.getText().substring(0,txtAr.getSelectionStart()));
         String                                                          strtmp2=new
String(txtAr.getText().substring(txtAr.getSelectionEnd(),txtAr.getText().length()));    //返回一个
新的 String，它包含此序列当前所包含的字符子序列。
         txtAr.setText(strtmp1+strtmp2);
     }
    }

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

```
    new Test4_12();
    }
}
```

【13】创建一个输入对话框，从对话框中输入文字，当按下“确定”按钮后，能在屏幕上显示输入的文字。

[解答]：Test4_13.java

```
    import java.awt.*;
    import java.awt.event.*;

    class Test4_13 extends Frame implements ActionListener{
    TextArea txtAr;
    Button btn1;
    public Test4_13(){
        super("对话框练习");
        setSize(300,400);
        setVisible(true);
        setLayout(new BorderLayout());

        btn1=new Button("打开对话框");
        txtAr=new TextArea(8,10);

        add("North",btn1);
        add("Center",txtAr);
        validate();

         addWindowListener(new WindowAdapter(){
                    public void windowClosing(WindowEvent we){
            System.exit(0);
        }}   );

        btn1.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e){
    DiaObj dia=new DiaObj(this,txtAr);   //声明和实例话对话框,传递 frame,和 txtar 参数
    }


    public static void main(String[] args){
       new Test4_13();
    }

   }
```

```
class DiaObj implements ActionListener {
  Frame fr=new Frame();
  TextArea txtAr2=new TextArea(8,10);
  TextField txtFl=new TextField(20);
  Button btnSend=new Button("发送文字");
  Dialog dia;

//初始化 DiaObj,并定义参数 fr,txtAr2
  public DiaObj(Frame fr,TextArea txtAr2){
        this.txtAr2=txtAr2;
        this.fr=fr;
        dia=new Dialog(fr,"传递消息对话框..",true);
        dia.setSize(200,100);
        dia.setLayout(new FlowLayout());
        dia.add(txtFl);
        dia.add(btnSend);

        btnSend.addActionListener(this);

        dia.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent we){
              dia.setVisible(false);   //退出对话框
        }}    );

        dia.validate();
        dia.setVisible(true);
  }

  public void actionPerformed(ActionEvent e){
        txtAr2.append(txtFl.getText()+"\n");
        txtFl.setText("");//清空内容
  }
}
```

第 5 章 Java swing 基础

【1】 应用 swing 组件，改写【例 4-4】的密码验证程序。

[解答]:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class XitiDaan5_1 extends JFrame implements ActionListener{
    JLabel jl;
    JTextField jt2;
    JPanel jp;
    TextField t1;
    JButton jb;
    XitiDaan5_1(){
        super("密码验证");
        setSize(800,600);
        setVisible(true);
        Container con=getContentPane();
        con.setLayout(new BorderLayout());
        jp=new JPanel();
        jl=new JLabel("请输入密码：");
        t1=new TextField(25);
        t1.setEchoChar('*');
        jt2=new JTextField(25);
        jb=new JButton("确定");
        jp.add(jl);
        jp.add(t1);
        jp.add(jb);
        jp.add(jt2);
        con.add(jp,BorderLayout.CENTER);
        jb.addActionListener(this);
        validate();
        }
    public void actionPerformed(ActionEvent e){
        if(t1.getText().equals("abc"))
jt2.setText("密码正确！！ ");
        else
jt2.setText("密码错误！！ ");
        }
    public static void main(String[] args){
new XitiDaan5_1();
}
```

}

程序运行结果如图 5.1 所示。

图 5.1

【2】将通讯录显示到一个表格中。

[解答]：

```
//  将通讯录显示到表格
 import javax.swing.*;
 import java.awt.*;
 import java.awt.event.*;
 class Tongxunlu_Table extends JFrame
 {
 public Tongxunlu_Table()
     {
      super("班级通讯录");
      String[] columnNames = {"姓名","性别","班级","固定电话","手机","地址"};
      Object[][] data = {
        {"张　三","男","040703","0595-22082642","13959941987","泉州"},
        {"李　四","男","040703","0595-22082643","13959941237","南安"},
        {"王　五","男","040703","0595-22082645","13959941987","泉州"},
        {"张　也","女","040703","0595-22082649","13959574487","福州"},
        {"李　名","男","040703","0595-22082644","13955421187","泉州"},
        {"张　涛","男","040703","0595-22082647","13959998887","厦门"},
        {"章　萍","女","040703","0595-22082640","13959941887","泉州"},
        {"谢少福","男","040703","0595-22082643","13959941987","漳州"},
        {"黄志强","男","040703","0595-22082641","13955647887","泉州"},
        {"张志晨","男","040703","0595-22082646","13988441987","龙岩"},
```

```
            {"张明美","女","040703","0595-22082638","13959988198","莆田"}
        };
    JTable table = new JTable(data,columnNames);
        table.setPreferredScrollableViewportSize(new Dimension(500,70));
        JScrollPane scrollPane = new JScrollPane(table);
        getContentPane().add(scrollPane, BorderLayout.CENTER);
        addWindowListener(new WindowAdapter() {
          public void windowClosing(WindowEvent e) {
            System.exit(0);
          }
        });
        pack();
    setSize(800,600);
        setVisible(true);
  }
  }
   //主类
  public class Xitidaan5_2
  {
    public static void main(String[] args)
      {
          Tongxunlu_Table frame = new Tongxunlu_Table();
      }
  }
```

程序运行结果如图 5.2 所示。

姓名	性别	班级	固定电话	手机	地址
张 三	男	040703	0595-22082642	13959941987	泉州
李 四	男	040703	0595-22082643	13959941237	南安
王 五	男	040703	0595-22082645	13959941987	泉州
张 也	女	040703	0595-22082649	13959574487	福州
李 名	男	040703	0595-22082644	13955421187	泉州
张 涛	男	040703	0595-22082647	13959998887	厦门
章 萍	女	040703	0595-22082640	13959941887	泉州
谢少福	男	040703	0595-22082643	13959941987	漳州
黄志强	男	040703	0595-22082641	13955647887	泉州
张志晨	男	040703	0595-22082646	13988441987	龙岩
张明美	女	040703	0595-22082638	13959988198	莆田

图 5.2

【3】 改进【例 5-11】，编写一个能动态改变树结点的程序。

[解答]:

```
/* "利用 TreeNode 构造动态树" */
import     javax.swing.*;
import     javax.swing.tree.*;
```

```
import java.awt.*;
import java.awt.event.*;
import java.awt.GridLayout;
import java.awt.Toolkit;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTree;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;
import javax.swing.tree.MutableTreeNode;
import javax.swing.tree.TreePath;
import javax.swing.tree.TreeSelectionModel;
import javax.swing.event.TreeModelEvent;
import javax.swing.event.TreeModelListener;

class TreeNodeChild extends DefaultMutableTreeNode{
  JPanel Parent;
  JRadioButton PartRadio;
  JLabel PartLabel;
  public TreeNodeChild(String Node){
        Parent = new JPanel();
        PartRadio = new JRadioButton();
        PartLabel = new JLabel();
        PartLabel.setText(Node);
        Parent.add(PartRadio);
        Parent.add(PartLabel);
        }
    }
class DynamicTree extends JPanel {
      TreeNodeChild rootNode;
      DefaultTreeModel treeModel;
      JTree tree;
      Toolkit toolkit = Toolkit.getDefaultToolkit();
      DynamicTree(){
        super(new GridLayout(1,0));
        TreeNodeChild rootNode = new TreeNodeChild("Root Node");
        treeModel = new DefaultTreeModel(rootNode);
        treeModel.addTreeModelListener(new MyTreeModelListener());

        tree = new JTree(treeModel);
        tree.setEditable(true);
```

```
tree.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
        tree.setShowsRootHandles(true);

        JScrollPane scrollPane = new JScrollPane(tree);
        add(scrollPane);
        }
    public void removeCurrentNode() {
        TreePath currentSelection = tree.getSelectionPath();
        if(currentSelection != null){
            DefaultMutableTreeNode
currentNode=(DefaultMutableTreeNode)(currentSelection.getLastPathComponent());
            MutableTreeNode parent=(MutableTreeNode)(currentNode.getParent());
              if(parent!=null){
                treeModel.removeNodeFromParent(currentNode);
                return;
                }
        }
        toolkit.beep();
    }

    public TreeNodeChild addObject(Object child){
        TreeNodeChild parentNode = null;
        TreePath parentPath = tree.getSelectionPath();

        if(parentPath==null){
            parentNode=rootNode;
            }else{
                parentNode = (TreeNodeChild)(parentPath.getLastPathComponent());
                }

            return addObject(parentNode, child, true);
        }

    public TreeNodeChild addObject(TreeNodeChild parent, Object child){
            return addObject(parent, child, false);
        }

    public TreeNodeChild addObject(TreeNodeChild parent,Object child, boolean
shouldBeVisible){
        TreeNodeChild childNode = new TreeNodeChild(child.toString());

        if(parent==null){
            parent = rootNode;
            }
```

```
            treeModel.insertNodeInto(childNode, parent, parent.getChildCount());
            if(shouldBeVisible){
              tree.scrollPathToVisible(new  TreePath(childNode.getPath()));
              }
            return  childNode;
            }}
class  MyTreeModelListener  implements  TreeModelListener  {
      public  void  treeNodesChanged(TreeModelEvent  e)  {
      DefaultMutableTreeNode  node;
      node = (DefaultMutableTreeNode)(e.getTreePath().getLastPathComponent());
        try{
            int  index  =  e.getChildIndices()[0];
            node = (DefaultMutableTreeNode)(node.getChildAt(index));
           }catch(NullPointerException  exc)  {}

        System.out.println("The  user  has  finished  editing  the  node.");
        System.out.println("New  value:  "  +  node.getUserObject());
       }
      public  void  treeNodesInserted(TreeModelEvent  e)  { }
      public  void  treeNodesRemoved(TreeModelEvent  e)  {}
      public  void  treeStructureChanged(TreeModelEvent  e)  { }
            }
public class XitiDaan5_3 extends JFrame implements  ActionListener{
    JButton jb1,jb2;
    JPanel jp2;
    // JPanel jp1;
    int  newNodeSuffix =1;
    String  ADD_COMMAND="添加";
    String  REMOVE_COMMAND="移除";
    DynamicTree  treePanel;
    XitiDaan5_3(String s){
    super(s);
    treePanel= new DynamicTree();
    // populateTree(treePanel);
    jb1=new JButton("添加");
    jb2=new JButton("移除");
    //jp1=new JPanel();
    jp2=new JPanel();
    Container con=getContentPane();
    con.setLayout(new BorderLayout());
    /* DefaultMutableTreeNode root=new DefaultMutableTreeNode("c:\\");//树的根节点。
    DefaultMutableTreeNode t1=new DefaultMutableTreeNode("备份资料");//节点。
```

```
    DefaultMutableTreeNode t2=new DefaultMutableTreeNode("Java 学习");//节点。
    DefaultMutableTreeNode t1_1=new DefaultMutableTreeNode("思维论坛精华帖子");
    DefaultMutableTreeNode t1_2=new DefaultMutableTreeNode("来往邮件");
    DefaultMutableTreeNode t2_1=new DefaultMutableTreeNode("视频教程");
    DefaultMutableTreeNode t2_2=new DefaultMutableTreeNode("Java3D");
    root.add(t1);root.add(t2);
    t1.add(t1_1);t1.add(t1_2);//t1_1,t1_2 成为 t1 的子节点。
    t2.add(t2_1);t2.add(t2_2);//t2_1,t2_2 成为 t2 的子节点。
    JTree tree =new JTree(root); //创建根为 root 的树。
    JScrollPane scrollpane=new JScrollPane(tree);
    jp1.add(scrollpane);*/
    jp2.add(jb1);jp2.add(jb2);
    treePanel.setPreferredSize(new    Dimension(300,    150));
    con.add(treePanel,BorderLayout.CENTER);
    // con.add(jp1,BorderLayout.NORTH);
    con.add(jp2,BorderLayout.SOUTH);
    setSize(800,600);
    setVisible(true);
    validate();
    jb1.setActionCommand("添加");
    jb1.addActionListener(this);
    jb2.setActionCommand("移除");
    jb2.addActionListener(this);
    addWindowListener(new WindowAdapter(){public void windowClosing(WindowEvent e)
      {System.exit(0);}
      });
    }
  public    void actionPerformed(ActionEvent e){
      String    command    =    e.getActionCommand();
  if (ADD_COMMAND.equals(command)){
      treePanel.addObject("New Node " + newNodeSuffix++);
      }else if(REMOVE_COMMAND.equals(command)) {
            treePanel.removeCurrentNode();
            }
    }

public static void main(String[] args){
    new XitiDaan5_3("利用 TreeNode 构造动态树");
    }
}
```

程序运行结果如图 5.3 所示。

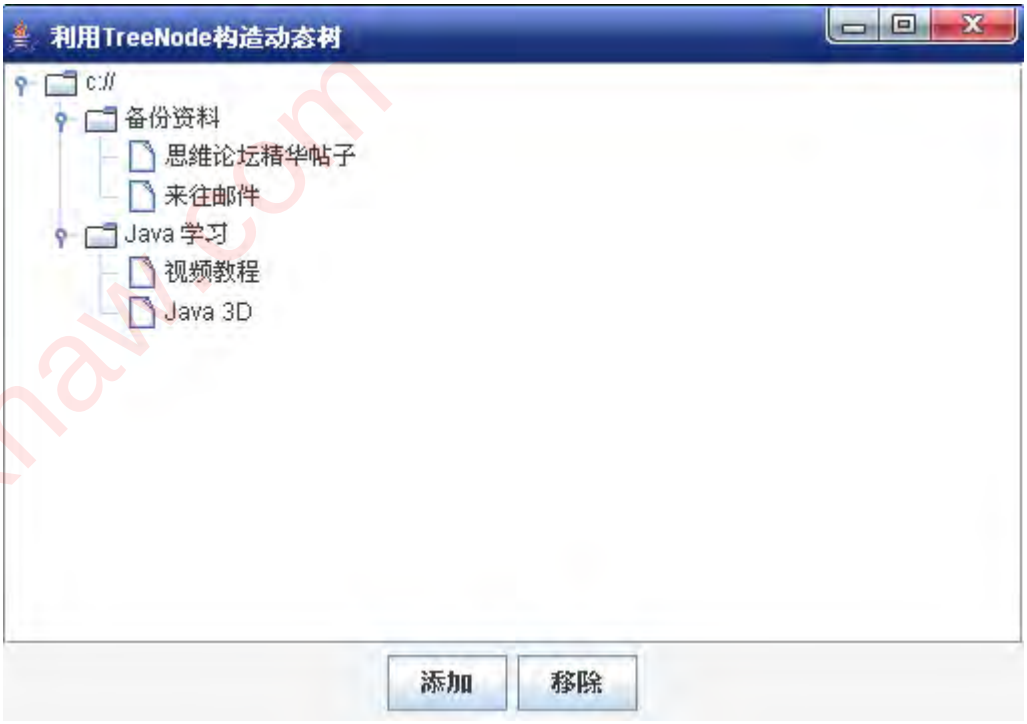


图 5.3

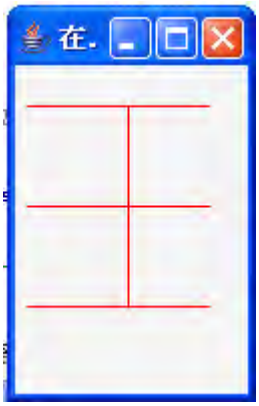
第六章 Java 图形与图像处理

【1】用绘制线段的方法输出一个红色的“王”字。

[解答]：代码如下：

```
import javax.swing.*;
import java.awt.*;
public class DrawWang extends JFrame {
    public DrawWang(){
        super("在窗体上绘制一个王字");
    }
    public void paint(Graphics g){
        g.setColor(Color.red);
        g.drawLine(10,50,100,50);
        g.drawLine(10,100,100,100);
        g.drawLine(10,150,100,150);
        g.drawLine(60,50,60,150);
    }
    public static void main(String[] args) {
        DrawWang dw = new DrawWang();
        dw.setSize(150,200);
        dw.setVisible(true);
    }
}
```

运行效果如下：



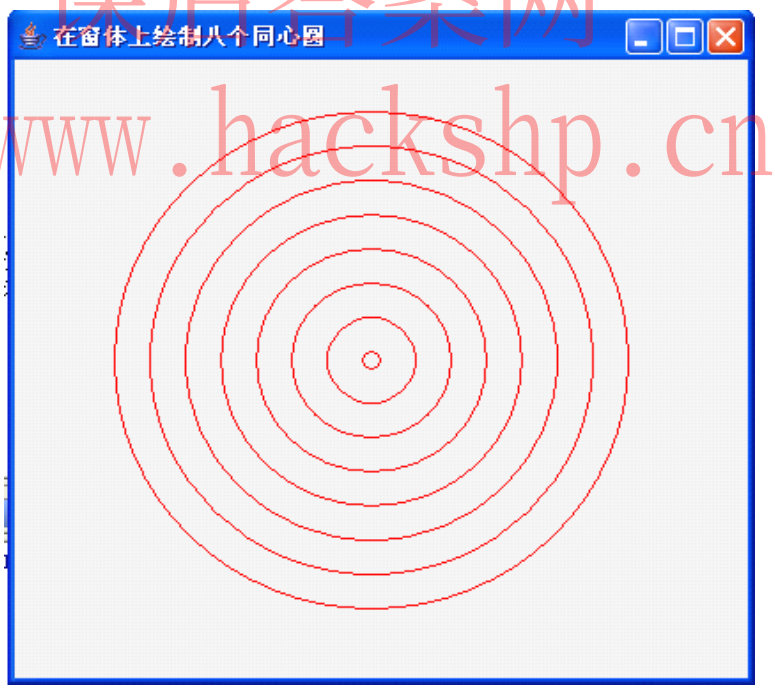
【2】编写一个程序绘制 8 个同心圆，各园相差 20 个像素点。

[解答]：代码如下：

```
import javax.swing.*;
import java.awt.*;
public class EightCircle extends JFrame {
    public EightCircle(){
        super("在窗体上绘制八个同心圆");
```

```
    }
    public void paint(Graphics g){
        g.setColor(Color.red);
        g.drawOval(200,200,10,10);
        g.drawOval(180,180,50,50);
        g.drawOval(160,160,90,90);
        g.drawOval(140,140,130,130);
        g.drawOval(120,120,170,170);
        g.drawOval(100,100,210,210);
        g.drawOval(80,80,250,250);
        g.drawOval(60,60,290,290);
    }
    public static void main(String[] args) {
        EightCircle ec = new EightCircle();
        ec.setSize(500,400);
        ec.setVisible(true);
    }
}
```

运行效果如下：



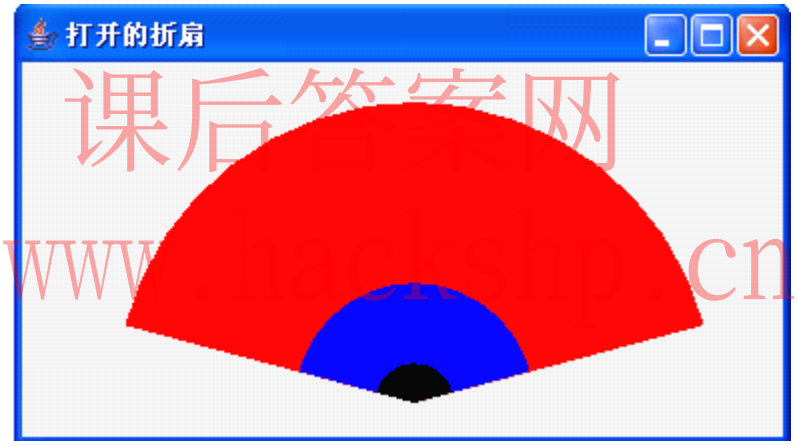
【3】编写一个程序绘制一把打开的折扇。

[解答]：代码如下：

```
import javax.swing.*;
import java.awt.*;
public class T5 extends JFrame {
    public T5(){
        super("打开的折扇");
```

```
    }
    public void paint(Graphics g){
        g.setColor(Color.red);
        g.fillArc(50,50,300,300,15,150);
        g.setColor(Color.BLUE);
        g.fillArc(140,140,120,120,15,150);
        g.setColor(Color.BLACK);
        g.fillArc(180,180,40,40,15,150);
    }
    public static void main(String[] args) {
        T5 ec = new T5();
        ec.setSize(500,400);
        ec.setVisible(true);
    }
}
```

运行效果如下：



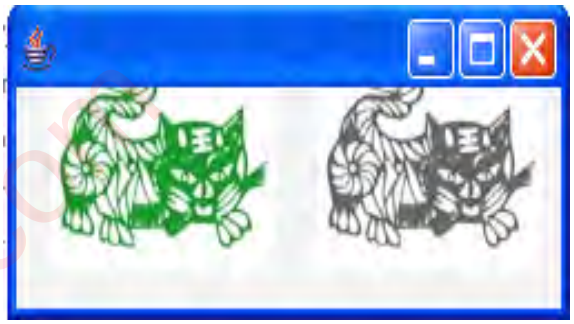
【4】改变一副彩色照片的颜色，使之成为黑白照片。

[解答]：代码如下：

```
import javax.swing.*;
import java.awt.*;
import java.awt.image.PixelGrabber;
import java.awt.image.ColorModel;
import java.awt.image.ImageProducer;
import java.awt.image.MemoryImageSource;
public class ConvertToBlack extends JFrame{
    private Image imgOriginal,imgChanged;
    private int[] pixels;
    public ConvertToBlack(){
        //load original image
        imgOriginal = new ImageIcon("src/images/swingo.jpg").getImage();
        int height = imgOriginal.getHeight(this);
        int width = imgOriginal.getWidth(this);
```

```
        pixels = new int[width * height];
        System.out.println(height);
        //grabber pixels from image to Array
        PixelGrabber grabber= new PixelGrabber(imgOriginal,0,0,width,height,pixels,0,width);
        try {
            grabber.grabPixels();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        //change the pixels
        ColorModel model = ColorModel.getRGBdefault();
        for (int i = 0; i < pixels.length; i++) {
            int pixel = pixels[i];
            int alpha = model.getAlpha(pixel);
            int gray = (int)(model.getRed(pixel) * 0.3
                    + model.getGreen(pixel) * 0.59
                    + model.getBlue(pixel) * 0.11);
            int red = gray;
            int green = gray;
            int blue = gray;
            pixels[i] = (alpha << 24) | (red << 16) | (green << 8) | blue;
        }
        //create new image
        ImageProducer producer = new MemoryImageSource(width,height,pixels,0,width);
        imgChanged = createImage(producer);
        setSize(300,300);
        setVisible(true);
    }
    public static void main(String[] args) {
        new ConvertToBlack();
    }
    public void paint(Graphics g){
        g.drawImage(imgOriginal,20,20,this);
        g.drawImage(imgChanged,140,20,this);
    }
}
```

运行效果如下：



【5】编写一段产生三维文字的程序。

[解答]：请先安装 Java 3D 类库，代码如下：

```
import com.sun.j3d.utils.universe.SimpleUniverse;
import com.sun.j3d.utils.universe.ViewingPlatform;
import com.sun.j3d.utils.behaviors.vp.OrbitBehavior;
import javax.media.j3d.*;
import javax.vecmath.Point3f;
import javax.vecmath.Point3d;
import javax.vecmath.Color3f;
import javax.vecmath.Vector3f;
import java.awt.*;
public class Text3DC {
    private String fontName = "default";
    private String textString = null;
    private SimpleUniverse u;
    private OrbitBehavior orbit;
    public BranchGroup createSceneGraph(){
        float sl = textString.length();
        BranchGroup objRoot = new BranchGroup();
        TransformGroup objScale = new TransformGroup();
        Transform3D t3d = new Transform3D();
         // 设置字体大小
        t3d.setScale(1.5/sl);
        objScale.setTransform(t3d);
        objRoot.addChild(objScale);
        TransformGroup objTrans = new TransformGroup();
        objScale.addChild(objTrans);
        //建立 3D 字
        Font3D f3d = new Font3D(new Font(fontName, Font.PLAIN, 2),
                        new FontExtrusion());
        Text3D txt = new Text3D(f3d,textString,new Point3f(-sl/2.0f, -1.f, -1.f));
        Shape3D sh = new Shape3D();
        Appearance app = new Appearance();
        Material mm = new Material();
        mm.setLightingEnable(true);
```

```
        app.setMaterial(mm);
        sh.setGeometry(txt);
        sh.setAppearance(app);
        objTrans.addChild(sh);
        //设置场景的范围
        BoundingSphere bounds=new BoundingSphere(new Point3d(0.0,0.0,0.0),100.0);
        //使物体旋转
       if (false){
          Transform3D yAxis = new Transform3D();
          Alpha rotationAlpha = new Alpha(-1, Alpha.INCREASING_ENABLE,
                                          0, 0, 4000, 0, 0, 0, 0, 0);
          RotationInterpolator rotator =
              new RotationInterpolator(rotationAlpha, objTrans, yAxis,
                                          0.0f, (float) Math.PI*2.0f);
          rotator.setSchedulingBounds(bounds);
          objTrans.addChild(rotator);
        }
        // 设置背景
        Color3f bgColor = new Color3f(0.15f, 0.15f, 0.5f);
        Background bgNode = new Background(bgColor);
        bgNode.setApplicationBounds(bounds);
        objRoot.addChild(bgNode);
        // 设置环境光源
        Color3f ambientColor = new Color3f(0.9f, 0.9f, 0.9f);
        AmbientLight ambientLightNode = new AmbientLight(ambientColor);
        ambientLightNode.setInfluencingBounds(bounds);
        objRoot.addChild(ambientLightNode);
        // 设置光源方向
        Color3f light1Color = new Color3f(1.0f, 1.0f, 0.9f);
        Vector3f light1Direction  = new Vector3f(1.0f, 1.0f, 1.0f);
        Color3f light2Color = new Color3f(1.0f, 1.0f, 0.9f);
        Vector3f light2Direction  = new Vector3f(-1.0f, -1.0f, -1.0f);
        DirectionalLight light1
              = new DirectionalLight(light1Color, light1Direction);
        light1.setInfluencingBounds(bounds);
        objRoot.addChild(light1);
        DirectionalLight light2
              = new DirectionalLight(light2Color, light2Direction);
        light2.setInfluencingBounds(bounds);
        objRoot.addChild(light2);
        return objRoot;
    }
    public Text3DC(){
        textString = "Java3D";
```

```
        BranchGroup scene = createSceneGraph();
        u = new SimpleUniverse();
        //在虚拟空间中添加鼠标行为
        ViewingPlatform viewingPlatform = u.getViewingPlatform();
        viewingPlatform.setNominalViewingTransform();
        //设置鼠标控制
        orbit = new OrbitBehavior();
        BoundingSphere bounds =
                new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
        orbit.setSchedulingBounds(bounds);
        viewingPlatform.setViewPlatformBehavior(orbit);
        u.addBranchGraph(scene);
    }
    public static void main(String[] args) {
        new Text3DC();
    }
}
```

运行效果如下：

第 7 章 多线程与异常处理

【1】java 为什么要引入线程机制，线程，程序和进程之间的关系是怎样的？
[解答]：Java 之所以引入线程机制是因为：线程间的通信非常简单且有效，上下文切换非常快，它们是同一个进程中的两部分之进行的切换，每个线程彼此独立执行，一个程序可以同时使用多个线程来完成不同的任务。

简而言之,一个程序至少有一个进程,一个进程至少有一个线程.线程的划分尺度小于进程，使得多线程程序的并发性高。另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动,进程是系统进行资源分配和调度的一个独立单位。线程是进程的一个实体,是 CPU 调度和分派的基本单位,它是比进程更小的能独立运行的基本单位.线程自己基本上不拥有系统资源,只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈),但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源.一个线程可以创建和撤销另一个线程;同一个进程中的多个线程之间可以并发执行。

进程和线程的主要差别在于它们是不同的操作系统资源管理方式。进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。

【2】线程有哪几种基本状态，试描述它们之间的转换图。
[解答]：
1) 新建:当一个 Thread 类或者其子类的对象被声明并创建时，新的线程对象处于新建状态，此时它已经有了相应的内存空间和其他资源。
2) 就绪:处于新建状态的线程被启动后，将进入线程队列排队等待 CUP 服务，这个时候具备了运行的条件，一旦轮到 CPU 的时候，就可以脱离创建它的主线程独立开始自己的生命周期。
3) 运行:就绪的线程被调度并获得 CUP 的处理边进入了运行状态，每一个 Thread 类及其子类的对象都有一个重要的 run()方法，当线程对象被调度执行的时候，它将自动调用本对象的 run()方法，从第一句代码开始执行。所以说对线程的操作应该写到 run()方法中。
4) 阻塞:一个正在执行的线程如果再某种情况下不能执行了.进入阻塞状态，这个时候它不能进入排队状态，只有引起了阻塞的原因消失的时候，线程才可以继续进入排队状态等待 CUP 处理。
5) 死亡:处于死亡状态的线程不具有继续执行的能力，线程死亡主要的原因是正常运行的线程完成了全部工作，即执行完了 run()方法，另外就是被提前强制的终止了。

【3】Runnable 接口有哪些抽象的方法？Thread 类有哪些主要域和方法

[解答]:

Runnable 接口就一个抽象的方法 run()

Thread 类主要的域有：

```
private char      name[];                                  名字。
private int       priority;                                优先级别
public final static int MIN_PRIORITY = 1;                  最小优先级定义
public final static int MAX_PRIORITY = 10;                 最大优先级定义
```

等。

主要的方法有：

```
public static native Thread currentThread()                返回当前线程
public static native void yield()停当前线程，让出其它的线程可以得以执行
public static native void sleep(long millis)               线程休眠一段时间
public synchronizæd void start()                           开始线程
public void run()                                          线程的主要运行方法
public final void suspend()                                线程挂起
private native void resume()                               线程挂起以后恢复执行
```

【4】创建线程有几种方式？为什么有时候必须采用其中种方式，试写出使用这种方式创建线程的一般模式。

[解答]：有两种方式，一种是实现 Runnable 接口，一种是继承 Thread 类。

因为 java 不能支持多继承，所以有的时候一个类如果已经继承了别的类，那他就不能用 Thread 类来继承这种方式来创建一下线程，所以就只能实现 Runnable 接口。

这种方式创建线程的一般模式是：

（1） 实现 Runnable 接口。

（2） 复写 run 方法。

（3） 新建线程对象的时候以这个对象的实例做为参数值。

（4） 调用线程类的 start()方法来启动线程。

【5】举例说明线程同步的概念。

[解答]：线程的同步问题主要是因为多个线程要同时使用某个临介资源的时候来造成数据的不正确产生的。如果把资源做成一个对象的话，这时为了正确的使用这个资源，在一个线程使用此对象的时候，此对象的其它的线程就不应该能访问这个资源，或以一个安全的方式来访问。这就是 java 的线程同步要解决的主要问题。

举例说明：在使用数据库的时候，如果一个用户正在对一条记录进行修改，而另一个用记也要对这条记录时行修改，这样就形成线程同步的问题。

【6】试用线程的方法编写两个 10*10 矩阵的相乘的计算程序，用 10 个线程完成结果矩阵每一行的计算。

[解答]:

```
/**
    * 两个 10*10 的矩阵相乘的程序
    */
public class TenThread {
```

```
//定义三个 1 0 1 10*10 的矩阵，c 是用来存放结果的
int[][] a = new int[10][10];

int[][] b = new int[10][10];

int[][] c = new int[10][10];

/**
 * 内部的线程类，用来实现矩阵的行列相乘
 */
class MultiThread extends Thread {
    int i = 0;

    //传入的参数表示行数
    public MultiThread(int i) {
        this.i = i;
    }

    public void run() {
        for (int k1 = 0; k1 < 10; k1++) {
            for (int k = 0; k < 10; k++) {
                c[i][k1] = c[i][k1] + a[i][k] * b[k][k1];
            }
        }
    }
}

public TenThread() {
    //赋值的数据
    int count = 0;
    //把矩阵赋值
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++) {
            a[i][j] = count;
            b[i][j] = count++;
        }
    }
}

public static void main(String[] args) {
    TenThread tt = new TenThread();
    for (int i = 0; i < 10; i++) {
        tt.new MultiThread(i).start();
    }
```

```
        //等待所有线程的执行完成
        try{
        Thread.sleep(1000);
        }catch(Exception e){
              e.printStackTrace();
        }
        //打印
        for(int i=0;i<10;i++){
              for(int j=0;j<10;j++){
                    System.out.print(tt.c[i][j]+"        ");
              }
              System.out.println("\n");
        }
   }
}
```

【7】编写一个龟兔赛跑的多线程程序，单击按键以后龟兔开始赛跑。

[解答]:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.applet.*;

public class Race extends JApplet implements Runnable {
  int x1, x2, y1, y2;
  //速度
  int d1 = 10, d2 = 5;

  Thread a = null;
  Thread b=null;

  public void init() {
        Container cp = getContentPane();
        x1 = 10;
        x2 = 10;
        y1 = 50;
        y2 = 100;
  }

  public void start() {
         a = new Thread(this);
         b= new Thread(this);
         a.start();
         b.start();
```

```
}

public void run() {
    while (true) {
        if(Thread.currentThread()==a){
        x1 += d1;
        if(x1==200){
            d1=0;
        }
        }else{
        x2 += d2;
        }
        if (x1 > 300||x2>300) {
            System.exit(0);
            a.stop();
            b.stop();
        }

        repaint();
        try {
            a.sleep(100);
        } catch (InterruptedException e) {
        }
    }
}

public void paint(Graphics g) {

    g.setColor(Color.gray);
    g.fillRect(0, 0, 400, 200);
    g.setColor(Color.red);
    g.drawLine(310, 0, 310, 150);
    g.setColor(Color.red);
    g.fillOval(x1, y1, 10, 10);
    g.setColor(Color.green);
    g.fillOval(x2, y2, 10, 10);
    g.drawString("红球是兔子，绿球是乌龟！  ", 50, 180);
}
}
```

Html 代码

```
<HTML>
<BODY>
<APPLET CODE="Race.class" WIDTH="520" HEIGHT="300">
```

```
</APPLET>
</BODY>
</HTML>
```

【8】编写一个程序，让一个小球在窗体中跳动，当撞到边缘时，则选择一个角度反弹回去

[解答]:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class BallJump extends JFrame {
  static int x = 30,y=30;
      //小球的反弹角度
  static int j=30;
  double k=Math.tan(3.14*j/180.0);
  static int maxx=0,maxy=0;

  Container c;

  public BallJump() {
        c = this.getContentPane();
        c.add(new panel());
        this.setSize(200, 200);
        this.setVisible(true);
        maxx=(int)this.getSize().getWidth();
        maxy=(int)this.getSize().getHeight();

  }

  public static void main(String s[]) {
        new BallJump();
  }

  class panel extends JPanel//在面板上画个小球
  {
        panel() {
              MoveThread t = new MoveThread();
              t.start();
        }

        public void paint(Graphics g) {
              g.clearRect(0, 0, this.getWidth(), this.getHeight());
              g.setColor(Color.red);
```

```
            g.fillOval(x * 2, y, 20, 20);
        }
    }

    class MoveThread extends Thread//写一个线程控制 X 坐标增加
    {
        boolean jup=false;
        //向右移
        double yu=0;
        double xr=0;
        public void run() {
            while (true) {
                try {
                    Thread.sleep(10);
                    repaint();

                    //到边开始反弹
                    if(jup==false){
                        y=y+1;
                    }

                    if(y==(BallJump.maxy-20*2)||jup==true){
                        x++;
                        y--;
                        yu=(BallJump.maxy-20*2)-y;
                        xr=x;
                        if(yu/xr>k){
                            y++;
                        }else if(yu/xr<k){
                            x--;
                        }
                        if(x>maxx)
                            Thread.currentThread().stop();
                        jup=true;
                    }

                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

第 8 章 输入/输出流

【1】简述 java 流的概念、特点、及表示

[解答]: Java 的流是一个比文件所包含范围更广的概念。流是一个可被顺序访问的数据序列，是对计算机输入数据和输出数据的抽象。

Java 中的流是用类来表示。

Java 流的特点：数据可以是未经加工的原始二进制数据，也可以是经一定编码处理后符合某种格式规定的特定数据，java 中的数据流有字节流和字符流之分。

【2】描述 java.io 包中输入/输出流的类层次结构。

[解答]:

以字节为导向的 stream

基类是 InputStream 和 OutputSteam

stream 代表的是任何有能力产出数据的数据源，或是任何有能力接收数据的接收源。在Java 的 IO 中，所有的 stream（包括 Input 和 Output stream）都包括两种类型：

a) input stream：

1) ByteArrayInputStream：把内存中的一个缓冲区作为 InputStream 使用

2) StringBufferInputStream：把一个 String 对象作为 InputStream

3) FileInputStream：把一个文件作为 InputStream，实现对文件的读取操作

4) PipedInputStream：实现了 pipe 的概念，主要在线程中使用

b) Out stream

1) ByteArrayOutputStream：把信息存入内存中的一个缓冲区中

2) FileOutputStream：把信息存入文件中

3) PipedOutputStream：实现了 pipe 的概念，主要在线程中使用

以 Unicode 字符为导向的 stream 包括下面几种类型：

a) Input Stream

1) CharArrayReader：与 ByteArrayInputStream 对应

2) StringReader：与 StringBufferInputStream 对应

3) FileReader：与 FileInputStream 对应

4) PipedReader：与 PipedInputStream 对应

b) Out Stream

1) CharArrayWrite：与 ByteArrayOutputStream 对应

2) StringWrite：无与之对应的以字节为导向的 stream

3) FileWrite：与 FileOutputStream 对应

4) PipedWrite：与 PipedOutputStream 对应

以字符为导向的 stream 基本上对有与之相对应的以字节为导向的 stream。两个对应类实现的功能相同，只是在操作时的导向不同。

【3】说明输入流，输出流的概念及作用。如何实现输入和输出流类的读写方法的传递。

[解答]: 就流的运行方向来说，流分为输入流和输出流，输入流将外部数据引入计算机。输出流是交数据引导到外部设备。

输入输出流读写方法的传递一般可以以一个字节缓冲数组做为中间的桥梁。

【4】解释字节流，字符流，字节文件输入流和字符文件输出流的含义。
[解答]：字节流：以二进制数据这最基本的数据表示方式的流。
字符流：按每 16 位的 Unicode 码来处理字符数据的流。
字节文件输入流：字节文件输入流是从字节输入流中继承而来的，它用于处理二进制的文件输入操作。
字符文件输出流：字符文件输出流是从字符输出流中继承而来的，它用于处理字符为操作单位的文件数据的输出。

【5】简述 File 类在文件管理中的作用与使用方法。
[解答]：作用：提供了描述文件和目录的操作与管理的方法，它不负责数据的输入，输出。专门用来管理磁盘文件与目录。
使用方法：
（1） 创建 File 类的对象。
（2） 以 getName()等方法来判断或获取文件或目录信息。
（3） 对文件及目录进行操作

【6】计算 Fibonacii 数列，a1=1,a2=1...an=an-1+an-2 即前两个数是 1,从 3 个数开始，每个数是前两个数的和，计算数列的前 20 项，并用字节文件流的方式输出到一个文件，要求每 5 项 1 行。
[解答]：

```
import java.io.File;
import java.io.FileOutputStream;

/**
 * 计算 Fibonacii 数列的前 20 项
 */
public class Fibonacii {
    //数列的长度
    int i = 0;

    int[] f = null;

    public Fibonacii(int i) {
        this.i = i;
    }

    /**
     * 得到数列的函数
     * @return int[]
     */
    public int[] getFibonacii() {
        if (i < 2) {
```

```
            return new int[] { 1, 1 };
        } else {
            f = new int[i];
            //给数列赋初值
            f[0] = 1;
            f[1] = 1;
            for (int k = 2; k < i; k++) {
                f[k] = f[k - 1] + f[k - 2];
            }
            return f;
        }
    }

    /**
     * 保存入文件
     * @param name
     */
    public void saveToFile(String name) {
        try {
            File file = new File(name);
            FileOutputStream fo = new FileOutputStream(file);
            //换行
            int l = '\n';
            for (int i = 0; i < 20; i++) {
                //每 5 个一行
                if (i != 0 && i % 5 == 0) {
                    fo.write(l);
                }
                fo.write(f[i]);

            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    //test and display
    public static void main(String[] args) {
        int[] fb20 = null;
        Fibonacii fb = new Fibonacii(20);
        fb20 = fb.getFibonacii();
        //打印
        for (int i = 0; i < 20; i++) {
```

```
            System.out.println(fb20[i]);
        }
        fb.saveToFile("D:\\a.dat");
    }
}
```

【7】利用文件输入／输出流类编程实现一个信函文件的显示与复制

[解答]:

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;

/**
 *文件的显示和复制
 */
public class FileDisplayAndCopy {
    //复制的文件
    File fileCopy;
    //读取的文件名
    String filename="";
    //用来存放数据的 StringBuffer;
    StringBuffer sb=new StringBuffer("");
    /**
     * @param fileCopy  拷贝的文件
     * @param filename  源文件
     */
    public FileDisplayAndCopy(String fileCopy,String filename){
        this.filename=filename;
        this.fileCopy=new File(fileCopy);
    }
    /**
     * 读出内容并显示
     */
    public void display(){

        try {
            FileReader fr=new FileReader(filename);
            BufferedReader br=new BufferedReader(fr);
            //读数据
            String str;
            while((str=br.readLine())!=null){
                sb.append(str);
```

```
                    sb.append('\n');
                }


            } catch (Exception e) {
                e.printStackTrace();
            }
            //显示
            System.out.println(sb.toString());
        }

        public void copy(){
            try {
                FileWriter fw=new FileWriter(fileCopy);
                BufferedWriter bw=new BufferedWriter(fw);
                //写数据流
                bw.write(sb.toString(),0,sb.toString().length());
                bw.flush();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        //test
        public static void main(String[] args){

            FileDisplayAndCopy fda=new FileDisplayAndCopy("d:\\a.txt","d:\\b.txt");
            fda.display();
            fda.copy();
        }
}
```

【8】建立一个文本文件，输入一段短文，编写一个程序，统计文件中字符的个数，并将结果写入另一个文件

[解答]:

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;

/**
 * 统计文件中字符的个数，并将结果写入另一个文件
 */
public class FileCharCounter {
```

```
//源文件和目的文件
File fileDec;
String src;
StringBuffer sb = new StringBuffer("");
public FileCharCounter(String dec, String src) {
    this.fileDec = new File(dec);
    this.src = src;
}
/**
 * 统计数目
 * @return
 */
public int count() {
    try {
        sb = new StringBuffer("");
        FileReader fr = new FileReader(src);
        BufferedReader br = new BufferedReader(fr);
        //读数据
        String str;
        while ((str = br.readLine()) != null) {
            sb.append(str);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return sb.toString().length();
}
/**
 * 写文件
 */
public void writeTo(){
    try {
        FileWriter fw=new FileWriter(fileDec);
        BufferedWriter bw=new BufferedWriter(fw);

        //写数据流
        String c=String.valueOf(count());
        bw.write(c);
        bw.flush();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
//test
```

```
    public static void    main(String[] args){
        FileCharCounter fda=new FileCharCounter("d:\\a.txt","d:\\b.txt");
        System.out.println(fda.count());
        fda.writeTo();

    }

}
```

【9】建立一个文本文件，输入学生 3 门课的成绩，编写一个程序，读入这个文件中的数据，输出每门课的成绩的最小值，最大值和平均值。

[解答]:

成绩.txt 文件

```
id#000001 e#98 m#76 p#76
id#000002 e#54 m#74 p#76
id#000003 e#98 m#73 p#78
id#000004 e#98 m#77 p#76
id#000005 e#92 m#45 p#76
id#000006 e#94 m#33 p#74
id#000007 e#98 m#88 p#76
id#000008 e#96 m#34 p#76
```



```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.StringTokenizer;

/**
 * 建立一个文本文件，输入学生 3 门课的成绩 读入这个文件中的数据，输出每门课的成绩的最小值，最大值和平均值。
 */
class Student {
    String id;
    //英语成绩
    float e;
    //数字成绩
    float m;
    //物理成绩
    float p;
    /**
```

```
     * @return Returns the e.
     */
    public float getE() {
        return e;
    }
    /**
     * @param e
     * The e to set.
     */
    public void setE(float e) {
        this.e = e;
    }

    /**
     * @return Returns the id.
     */
    public String getId() {
        return id;
    }
    /**
     * @param id
     * The id to set.
     */
    public void setId(String id) {
        this.id = id;
    }
    /**
     * @return Returns the m.
     */
    public float getM() {
        return m;
    }
    /**
     * @param m
     * The m to set.
     */
    public void setM(float m) {
        this.m = m;
    }
    /**
     * @return Returns the p.
     */
    public float getP() {
        return p;
```

```
    }
    /**
     * @param p
     * The p to set.
     */
    public void setP(float p) {
        this.p = p;
    }
}

public class CountS {
    //文件路径
    String filepath;

    //list 用来存放学生数据
    List list = new ArrayList();

    public CountS(String str) {
        filepath = str;
        init();
    }

    //初始化操作，用来读数据和解析数据放入相应的对象中
    public void init() {
        try {
            FileReader fr = new FileReader(filepath);
            BufferedReader br = new BufferedReader(fr);
            //读数据
            String str;
            while ((str = br.readLine()) != null) {
                Student s = new Student();
                //解析数据
                StringTokenizer st = new StringTokenizer(str.toString(), " ");
                //前面是标号，后面是数据
                String first = "";
                String data = "";
                String elem = st.nextToken();
                //id 的解析
                first = elem.split("#")[0];
                data = elem.split("#")[1];
                s.setId(data);
                //英语成绩
                elem = st.nextToken();
                first = elem.split("#")[0];
```

```
                data = elem.split("#")[1];
                s.setE(Float.valueOf(data).floatValue());
                //数学成绩
                elem = st.nextToken();
                first = elem.split("#")[0];
                data = elem.split("#")[1];
                s.setM(Float.valueOf(data).floatValue());
                //物理成绩
                elem = st.nextToken();
                first = elem.split("#")[0];
                data = elem.split("#")[1];
                s.setP(Float.valueOf(data).floatValue());
                //加入 list
                list.add(s);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
}

public void countAnddisplay(){

    //0 是最小值，1,最大值 2 是平均分
    float[] e=new float[]{100,0,0};
    float[] m=new float[]{100,0,0};
    float[] p=new float[]{100,0,0};

    for (Iterator it = list.iterator(); it.hasNext();) {
        Student ele = (Student) it.next();
        //英语
        if(e[0]>ele.getE()){
            e[0]=ele.getE();
        }
        if(e[1]<ele.getE()){
            e[1]=ele.getE();
        }
        e[2]+=ele.getE();
        //数学
        if(m[0]>ele.getM()){
            m[0]=ele.getM();
        }
        if(m[1]<ele.getM()){
            m[1]=ele.getM();
```

```
                }
                m[2]+=ele.getM();
                //物理
                if(p[0]>ele.getP()){
                    p[0]=ele.getP();
                }
                if(p[1]<ele.getP()){
                    p[1]=ele.getP();
                }
                p[2]+=ele.getP();
            }
            //平均分
            e[2]=e[2]/list.size();
            m[2]=m[2]/list.size();
            p[2]=p[2]/list.size();
            //打印
            System.out.println("英语最小值:"+e[0]+" ");
            System.out.println("英语最大值:"+e[1]+" ");
            System.out.println("英语平均分:"+e[2]+" ");
            System.out.println("数学最小值:"+m[0]+" ");
            System.out.println("数学最大值:"+m[1]+" ");
            System.out.println("数学平均分:"+m[2]+" ");
            System.out.println("物理最小值:"+p[0]+" ");
            System.out.println("物理最大值:"+p[1]+" ");
            System.out.println("物理平均分:"+p[2]+" ");
        }
        public static  void main(String[] args){
            CountS cs=new CountS("d:/成绩.txt");
            cs.countAnddisplay();
        }

}
```

【10】对象流的作用是什么。

[解答]：可以将对象作为一个整体通过对象流进行传输和存储。

【11】编写程序，保存一个文本对象并检索对象的数据。

[解答]：

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
```

```
/**
 * 保存一个文本对象并检索对象的数据
 */

public class Contex implements Serializable {
    //其中的两个文本对象
    String text1 = "";

    String text2 = "";

    /**
     * @return Returns the text1.
     */
    public String getText1() {
        return text1;
    }

    /**
     * @param text1
     *                The text1 to set.
     */
    public void setText1(String text1) {
        this.text1 = text1;
    }

    /**
     * @return Returns the text2.
     */
    public String getText2() {
        return text2;
    }

    /**
     * @param text2
     *                The text2 to set.
     */
    public void setText2(String text2) {
        this.text2 = text2;
    }

    //for test
    public static void main(String[] args) {
        Contex c = new Contex();
```

```
            c.setText1("this is text1");
            c.setText2("this is test2");
            //将对象保存入文件
            File file = new File("d:/temp.dat");
            try {
                FileOutputStream fo = new FileOutputStream(file);
                ObjectOutputStream oo = new ObjectOutputStream(fo);
                oo.writeObject(c);
            } catch (Exception e) {
                e.printStackTrace();
            }
            //将对象从文件中取出，并操作其数据
            Contex d=null;
            try {
                FileInputStream fo = new FileInputStream(file);
                ObjectInputStream oo = new ObjectInputStream(fo);
                d=(Contex)oo.readObject();
                System.out.println(c);
                System.out.println(d);
                System.out.println(d.getText1());
                System.out.println(d.getText2());
            } catch (Exception e) {
                e.printStackTrace();
            }

        }
}
```

【12】利用 File 类的 delete()方法，编写程序，删除某一个指定文件。

[解答]:

```
import java.io.File;

/**
 *利用 File 类的 delete()方法，编写程序，删除某一个指定文件。
 */
public class FileDelete {
    File file=null;
    public FileDelete(String filename){
        file=new File(filename);
    }
    //delete 方法
    public void deleteFile(){
        file.delete();
    }
```

```
    //test
    public static void main(String[] args) {
        FileDelete f=new FileDelete("d:/temp.dat");
        f.deleteFile();
    }
}
```

【13】改写例 8－16 使之能打开一个文件对话框，从而播放选取的音频文件。

[解答]:

```
import java.awt.Button;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;

import javax.swing.JFileChooser;

import sun.audio.AudioPlayer;
import sun.audio.AudioStream;

class Sound
{
 FileInputStream file;
 BufferedInputStream buf;
 File filename;
  public Sound(File filename)
   {
    try
      {
        file=new FileInputStream(filename);
        buf=new BufferedInputStream(file);
        AudioStream audio=new AudioStream(buf);
        AudioPlayer.player.start(audio);
      }
    catch (Exception e) {
        e.printStackTrace();
    }
   }
}
```

```
public class Example8_16 extends Frame implements ActionListener
{
      File filename;
      Button btn;
      Button btn2;
     Example8_16()
      {
         super("音频播放器");
         setBounds(300,300,200,100);
         setVisible(true);
         btn=new Button("播放");
         btn2=new Button("选择文件");
         setLayout(new FlowLayout());
         add(btn);
         add(btn2);
         btn.addActionListener(this);
         btn2.addActionListener(this);
         validate();
         addWindowListener(new WindowAdapter()
             {   public void windowClosing(WindowEvent e)
                   { System.exit(0);  }
             });
      }

  public void actionPerformed(ActionEvent e)
     {
       if(e.getSource()==btn){
       Sound play = new Sound(this.filename);
       }else if(e.getSource()==btn2){
           JFileChooser fc = new JFileChooser(new File("."));
           if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
                filename = fc.getSelectedFile();
           }
       }
     }
      public static void main(String[] args)
      {
            new Example8_16();
      }
}
```

第 9 章 网络通信

【1】java 提供了哪几种网络通信模式？
[解答]：基于 TCP / IP 协议的面向连接的通信模式，基于 UDP 协议的面向无连接的通信模式。

【2】java 的套接字网络通信方式分为哪几种？
[解答]：
基于 TCP/IP 协议：客户端套接字，服务器端套接字。
基于 UDP 协议：用户数据报套接字，广播数据报套接字。

【3】什么是 socket，怎样建立 socket 连接？建立连接时，客户端和服务器端有什么不同？
[解答]：
Socket 就是套接字，是 IP 地址和端口号的组合。
当两个网络程序需要通信时，它们可以通过使用 Socket 类建立套接字连接。
服务器端建立 Socket 以后，要执行一个等待的方法，以方便客户端来连接。
客户端建立 Socket 以后即与服务器端进行连接，成功握手以后，双方产生同样的 Socket 对象。

【4】请列举常用的协议及其端口号。
[解答]：

ftp	21/tcp
telnet	23/tcp
smtp	25/tcp
http	80/tcp
pop3	110/tcp
snmp	161/udp
https	443/tcp
https	443/udp
pop3	110/tcp

【5】试描述用 Socket 建立连接的基本程序框架。
[解答]：
（1） 客户端建立套接字对象，指定服务器 IP 和端口号。
（2） 服务器端建立套接字，并指定端口号。
（3） 服务器端监听本机的端口的状态：执行 accept()方法。
（4） 客户端程序在对象产生以后以及，服务器端的程序监听到有连接以后都会产生一个 Socket 类的实例。
（5） 对这两个实例中对应的输入流和输出流进行操作，即完成通信过程。

【6】说明客户端如何与服务器端进行连接。
[解答]：

TCP / IP 的方式是：客户端产生 Socket 对象的同时产生与对应端口号的服务器连接的动作。
UDP 数据报的方式是：客户端建立 DatagramSocket 对象，建立报文 DatagramPacket 对象，并指定发送的 IP 地址，调用 socket 对象的 send 方法进行连接并发送数据。

【7】说明客户端如何从服务器读一行文本。
[解答]:

TCP / IP 的方式是：客户端与服务器端成功握手以后，从 Socket 中得到数据输入流并用相应的包装器进行包装，以输入流中相应的 read 方法来读取一行的文本。
UDP 数据报的方式是：建立数据报的 DatagramSocket 对象以后，调用 DatagramSocket 对象的 receive 方法来等待服务器的数据到来，接收到数据以后用 DatagramPacket 对象的 getData 方法来将接收到的数据提取出来。

【8】说明服务器如何将数据发送到客户端。
[解答]:

TCP / IP 的方式是：客户端与服务器端成功握手以后，从 Socket 中得到数据输出流并用相应的包装器进行包装，用并向客户端调用相应的 write 方法来发送数据。

UDP 数据报的方式是：创建数据报文 DatagramSocket 对象，调用 DatagramSocket 对象的 receive 方法来等待客户端的请求到来。从到来的 DatagramPacket 包中得到地址和端口号，建立数据报文对象，发送数据报。

【9】采用套接字的连接方式编写一个程序，允许客户向服务器提出一个名字，如果这个文件存在，就把文件内容发送给客户，否则回答文件不存在。
[解答]:

```
//客户端
import java.awt.Button;
import java.awt.FlowLayout;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedInputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.Socket;

import javax.swing.JFrame;


public class TranFileClient extends JFrame implements ActionListener {
    Button btn;
```

```
TextField tf;

public TranFileClient(){
      //布局
         super("要接收的文件名");
         setBounds(400,400,300,100);
         setVisible(true);
         btn=new Button("接收");
         tf=new TextField(7);
         setLayout(new FlowLayout());
         add(tf);
         add(btn);

         btn.addActionListener(this);

         validate();
         addWindowListener(new WindowAdapter()
             {   public void windowClosing(WindowEvent e)
                 { System.exit(0);   }
             });
}
public static void main(String[] args){
      new TranFileClient();

}
/* (non-Javadoc)
 * @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
 */
public void actionPerformed(ActionEvent arg0) {
      try {
            Socket st = new Socket("localhost",3997);
            //给服务器文件名
            DataOutputStream outServ =new DataOutputStream( st.getOutputStream());
            outServ.writeUTF(tf.getText().trim());
            //读服务器的消息
            FileOutputStream out = new FileOutputStream(tf.getText().trim());
            InputStream netin = st.getInputStream();
            InputStream in = new DataInputStream(new BufferedInputStream(netin));
            System.out.println("geting data.....");
            byte[] buf=new byte[2048];
            //
```

```
                int num = in.read(buf);
                while(num != -1){
                    out.write(buf, 0, num);
                    num = in.read(buf);
                }
                //取出服务器的状态
                String str=new String(buf);
                if(TranFileServer.PROP.equals(str.trim())){
                    tf.setText("文件没有找到！");
                }
                in.close();
                out.close();
                outServ.close();
                System.out.println("recieved over.....");
            } catch (IOException e) {
                e.printStackTrace();
            }

        }
}
```


服务器

```
import java.net.ServerSocket;
import java.net.Socket;
import java.io.*;

public class TranFileServer {
    //查找的文件的目录
    final static String PATH = "d:/src";
    final static String PROP = "NO";
    boolean isExist = false;
    public TranFileServer() {
        init();
    }

    private void init() {
        ServerSocket server;
        Socket st;
        String s = null;

        try {
            server = new ServerSocket(3997);
            System.out.println("waiting client connect");
            st = server.accept();
```

```
            //读客户端来的文件名
            try {
                DataInputStream in = new DataInputStream(st.getInputStream());
                while (true) {
                    s = in.readUTF();
                    if (s != null)
                        break;
                }
                System.out.println(s);
            } catch (IOException e) {
                System.out.println("ERRO:" + e);
                throw e;
            }

            doCheck(s);
            if (isExist) {
                FileInputStream fos = new FileInputStream(PATH+"/"+s);
                OutputStream netout = st.getOutputStream();
                DataOutputStream doc = new DataOutputStream(
                new BufferedOutputStream(netout));
                System.out.println("sending data....");
                byte[] buf = new byte[2048];
                int num = fos.read(buf);
                while (num != -1) {
                    doc.write(buf, 0, num);
                    doc.flush();
                    num = fos.read(buf);
                }
                fos.close();
                doc.close();
            }else{
                OutputStream netout = st.getOutputStream();
                DataOutputStream doc = new DataOutputStream(
                new BufferedOutputStream(netout));
                doc.write(PROP.getBytes());
                doc.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    /**
```

```
     * 文件是否存在判断
     *
     * @param s
     * @return
     */
    private void doCheck(String s) {
        boolean isExist;
        //文件是否存在
        File file = new File(PATH);
        String[] filenames = file.list();
        for (int i = 0; i < filenames.length; i++) {
            String str = filenames[i];
            if (str.equals(s)) {
                this.isExist = true;
                return;
            }
        }
    }

    public static void main(String[] args) {
        new TranFileServer();
    }
}
```


【10】写出使用多线程使得一个服务器同时为多个客户程序服务的基本框架。

[解答]:

```
/**
 * 多客户端一个服务器的例子
 */
import java.net.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class C_client extends Frame implements ActionListener {
    TextArea txt1;

    Button btn;

    Panel p;

    int port;

    DataInputStream in = null; //数据输入流
```

```
DataOutputStream out = null; //数据输出流

Socket c_socket; //套接字

InputStream in_data; //接收到的输入流

OutputStream out_data; //发送的输出流

String str; //存放接收的数据

int i = 0;

C_client() {
    super("客户端");
    setSize(300, 200);
    setVisible(true);
    txt1 = new TextArea(5, 4);
    add(txt1, BorderLayout.CENTER);
    p = new Panel();
    add(p, BorderLayout.NORTH);
    btn = new Button("连接");
    p.add(btn);
    btn.addActionListener(this);
    validate();
}

public static  void main(String[] args) {
    new C_client();
}

public void actionPerformed(ActionEvent eee) {
    try {
        c_socket = new Socket("127.0.0.1", 4321);

    } catch (IOException e) {
        e.printStackTrace();
    }

    try {
        in_data = c_socket.getInputStream();

        out_data = c_socket.getOutputStream();
        in = new DataInputStream(in_data);
```

```
            out = new DataOutputStream(out_data);

            int p1 = c_socket.getPort();
            int p2 = c_socket.getLocalPort();

            txt1.append("获取到对方的端口号：    " + p1 + "\n");
            txt1.append("本机的端口号：    " + p2 + "\n");

        } catch (IOException e) {
            e.printStackTrace();
        }

        try {

            str = in.readUTF();
            txt1.append("客户收到：" + str + "\n");
            if (i > 10) {
                out.writeUTF("end");
                c_socket.close();
                System.exit(0);
            }//发出 end 信息
            else {
                out.writeUTF("I am Client");
                i++;
            }

        } catch (IOException e) {
            System.out.println("ddd");
        }

    }

}


/**
 * 多客户端一个服务器的例子
 */
import java.net.*;import java.io.*;
import java.awt.*;import java.awt.event.*;
class S_server extends Frame implements ActionListener,Runnable
{
    ServerSocket        s_socket;   //服务器端套接字
    Socket              c_socket;   //套接字
```

```
    DataInputStream    in=null;     //数据输入流
    DataOutputStream out=null;    //数据输出流
    InputStream          in_data;     //接收到的输入流
    OutputStream         out_data;   //发送的输出流
    int                  i=0;          //计数(连接的客户数)
    String               str;
    TextArea             txt1;
    Button               btn;
    Panel                p;
S_server()
  {
     super("server");
     setSize(300,200);
     setVisible(true);
     txt1=new TextArea(5,4);
     add(txt1,BorderLayout.CENTER);
     p=new Panel();
     add(p,BorderLayout.NORTH);
     btn=new Button("开始监听端口");
     p.add(btn);
     validate();
     btn.addActionListener(this);
  }
 public void actionPerformed(ActionEvent eee)
  {
    try{
          s_socket=new ServerSocket(4321);
          while(true)
            {
            c_socket=s_socket.accept();
             Thread t=new Thread(this);
             t.start();
             i++;
            }
        }catch(IOException e){   }
  }
  //线程
 public void run()
  {
    try {
        while(true)
         {
          in_data=c_socket.getInputStream();
          out_data=c_socket.getOutputStream();
```

```
                in=new DataInputStream(in_data);
                out=new DataOutputStream(out_data);
                out.writeUTF("Hello,我是服务器");
                str=in.readUTF();
                if (str.equals("end"))
                   {//接收到 end 信息，则断开连接
                    in.close();
                   out.close();
                    c_socket.close();
                    }
                txt1.append("第"+i+"个客户发来："+str+"\n");
                Thread.sleep(200);
               } //while_end
            }
            catch(IOException e){
              e.printStackTrace();
              }
            catch(Exception ee){
            ee.printStackTrace();
            }   //Thread_catch
        }
  public static  void main(String[] args)
    {
       new    S_server();
    }
}
```

【11】写出一个客户同时有多个服务器为他提供服务的基本框架。

[解答]:

```
/**
 * 一个客户端多个服务器的例子
 */

import java.net.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class C_client1  extends Frame implements ActionListener, Runnable  {
        TextArea txt1;
        Button  btn;
        Panel p;
        int port;
        DataInputStream in = null; //数据输入流
```

```
DataOutputStream out = null; //数据输出流
//套接字
InputStream in_data; //接收到的输入流
OutputStream out_data; //发送的输出流
String str; //存放接收的数据

String[] IP = new String[] { "127.0.0.1", "127.0.0.2", "127.0.0.3" }; //服务器的 IP

int IPi = 0; //IP 的下标

C_client1() {
    super("客户端");
    setSize(300, 200);
    setVisible(true);
    txt1 = new TextArea(5, 4);
    add(txt1, BorderLayout.CENTER);
    p = new Panel();
    add(p, BorderLayout.NORTH);
    btn = new Button("连接");
    p.add(btn);
    btn.addActionListener(this);
    validate();
}

public static void main(String[] args) {
    new C_client1();
}

public void actionPerformed(ActionEvent eee) {
    Thread t1 = new Thread(this);
    t1.start();
    Thread t2 = new Thread(this);
    t1.start();
    Thread t3 = new Thread(this);
    t1.start();
}

public void run() {
    Socket c_socket = null;
    try {
        c_socket = new Socket(IP[IPi++], 4321);

    } catch (IOException e) {
        e.printStackTrace();
```

```
            }

            try {
                in_data = c_socket.getInputStream();

                out_data = c_socket.getOutputStream();
                in = new DataInputStream(in_data);
                out = new DataOutputStream(out_data);

                String p1 = c_socket.getInetAddress().getHostAddress();
                int p2 = c_socket.getLocalPort();

                txt1.append("获取到对方的 IP：      " + p1 + "\n");
                txt1.append("本机的端口号：      " + p2 + "\n");
                str = in.readUTF();
                txt1.append("客户收到：" + str + "\n");
                out.writeUTF("I am Client");

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

/**
 * 一个客户端多个服务器的例子
 */

import java.net.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class S_server1 extends Frame implements ActionListener{
    ServerSocket s_socket; //服务器端套接字
    Socket c_socket; //套接字
    DataInputStream in = null; //数据输入流
    DataOutputStream out = null; //数据输出流
    InputStream in_data; //接收到的输入流
    OutputStream out_data; //发送的输出流
    int i = 0; //计数(连接的客户数)
    String str;
```

```
    TextArea txt1;
    Button btn;
    Panel p;

    S_server1() {
        super("server");
        setSize(300, 200);
        setVisible(true);
        txt1 = new TextArea(5, 4);
        add(txt1, BorderLayout.CENTER);
        p = new Panel();
        add(p, BorderLayout.NORTH);
        btn = new Button("开始监听端口");
        p.add(btn);
        validate();
        btn.addActionListener(this);
    }

    public void actionPerformed(ActionEvent eee) {
        try {
            s_socket = new ServerSocket(4321);
            c_socket = s_socket.accept();
            in_data = c_socket.getInputStream();
            out_data = c_socket.getOutputStream();
            in = new DataInputStream(in_data);
            out = new DataOutputStream(out_data);
            out.writeUTF("Hello,我是服务器");
            str = in.readUTF();
            in.close();
            out.close();
            c_socket.close();
        } catch (IOException e) {
        }
  //Thread_catch
    }

    public static void main(String[] args) {
        new S_server();
    }
}
```

第 10 章 Java 数据库连接

【1】 试述 JDBC 提供了哪几种连接数据库的方法。

[解答]：JDBC 连接数据库的方法取决于 JDBC 驱动程序类型，Java 定义了 4 种 JDBC 驱动程序类型：

(1) JDBC-ODBC 桥驱动程序

JDBC-ODBC 桥接器负责将 JDBC 转换为 ODBC，用 JdbcOdbc.Class 和一个用于访问 ODBC 驱动程序的本地库实现的。这类驱动程序必须在服务器端安装好 ODBC 驱动程序，然后通过 JDBC-ODBC 的调用方法，进而通过 ODBC 来存取数据库。

(2) Java 到本地 API

这种类型的驱动程序是部分使用 Java 语言编写和部分使用本机代码编写的驱动程序，这类驱动程序也必须在服务器端安装好特定的驱动程序，如 ODBC 驱动程序，然后通过桥接器的转换，把 Java API 调用转换成特定驱动程序的调用方法，进而操作数据库。

(3) 网络协议搭配的 Java 驱动程序

这种驱动程序将 JDBC 转换为与 DBMS 无关的网络协议，这种协议又被某个服务器转换为一种 DBMS 协议。这种网络服务器中间件能够将它的纯 Java 客户机连接到多种不同的数据库上。所用的具体协议取决于提供者。

（4）本地协议纯 Java 驱动程序

这种类型的驱动程序将 JDBC 访问请求直接转换为特定数据库系统协议。不但无须在使用者计算机上安装任何额外的驱动程序，也不需要在服务器端安装任何中间程序，所有对数据库的操作，都直接由驱动程序来完成。

【2】 SQL 语言包括哪几种基本语句来完成数据库的基本操作。

[解答]：SQL 语言包括以下 6 种基本语句来完成数据库的基本操作：

(1)select 语句：用来对数据库进行查询并返回符合用户查询标准的结果数据。

(2)create table 语句：用来建立新的数据表。

(3)insert 语句：向数据表中插入或添加新的数据行。

(4)update 语句：更新或修改符合规定条件的记录。

(5)delete 语句：删除数据表中的行或记录。

(6)drop table 语句：删除某个数据表以及该表中的所有记录。

【3】 Statement 接口的作用是什么？

[解答]：Statement 接口用于执行静态 SQL 语句并返回它所生成结果的对象。在默认情况下，同一时间每个 Statement 对象在只能打开一个 ResultSet 对象。因此，如果读取一个 ResultSet 对象与读取另一个交叉，则这两个对象必须是由不同的 Statement 对象生成的。如果存在某个语句的打开的当前 ResultSet 对象，则 Statement 接口中的所有执行方法都会隐式关闭它。

【4】 ExecuteQuery()的作用是什么？

[解答]：ExecuteQuery()方法执行给定的 SQL 语句，返回单个 ResultSet 对象。发送给数据库的 SQL 语句，通常为静态 SQL SELECT 语句，返回包含给定查询所生成数据的 ResultSet 对象。

【5】 试述 DriverManager 对象建立数据库连接所用的几种不同的方法。

[解答]：DriverManager 对象建立数据库连接的方法有以下几种：

(1) static Connection getConnection(String url)： 使用指定的数据库 URL 创建一个连接。

(2) static Connection getConnection(String url, Properties info)：使用指定的数据库 URL 和相关信息（用户名、用户密码等属性列表）来创建一个连接，使DriverManager 从注册的 JDBC 驱动程序中选择一个适当的驱动程序。

(1)static Connection getConnection(String url, String user,String password)：使用指定的数据库 URL、用户名和用户密码创建一个连接，使 DriverManager 从注册的 JDBC 驱动程序中选择一个适当的驱动程序。

(2)static Driver getDriver(String url)：定位在给定 URL 下的驱动程序，让 DriverManager 从注册的 JDBC 驱动程序中选择一个适当的驱动程序。

【6】 编写一个应用程序，实现可以从一个数据库的某个表中查询一个列的所有信息。

[解答]：//英汉词典的应用示例，程序代码如下：

```
import java.sql.*;
public class GetColumnAllData{
public static void main(String args[]){
String cname,ename;
Connection Con=null;
Statement Stmt=null;
try{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");}
catch(ClassNotFoundException e){}
try{
Con=DriverManager.getConnection("jdbc:odbc:test","gxy","ookk");
Stmt=Con.createStatement();
ResultSet rs=Stmt.executeQuery("SELECT * FROM cidian ");
System.out.println("表中单词列的所有信息为：");
while (rs.next())
{ename=rs.getString("单词"); cname=rs.getString("解释");
System.out.println(ename);
}
Con.close();
}
catch(SQLException ee) {}
}
}
```

【7】编写一英汉字典程序，具有查询、添加、修改、删除等功能。

[解答]：程序代码如下：

```
import java.awt.*;
import java.sql.*;
import java.awt.event.*;
import javax.swing.*;
```

```
class DataWindow extends Frame implements ActionListener
{ TextField  待查英文单词_文本条,汉语解释_文本条,
更新英文单词_文本条,更新汉语解释_文本条,
填加英文单词_文本条,填加汉语解释_文本条,
删除英文单词_文本条,删除汉语解释_文本条;
Button  查询按钮,更新按钮,填加按钮,删除按钮,清空按钮;
int  查询记录=0;
Connection Con=null;Statement Stmt=null;
DataWindow()
{ super("英汉小词典");
  setBounds(150,150,300,150);
  setVisible(true);setLayout(new GridLayout(4,1));
  try{Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");}
  catch(ClassNotFoundException e){}
  try{
  Con=DriverManager.getConnection("jdbc:odbc:test","gxy","ookk");
  Stmt=Con.createStatement();
   }
   catch(SQLException ee) {}
   待查英文单词_文本条=new TextField(16);
   汉语解释_文本条=new TextField(16);
   更新英文单词_文本条=new TextField(16);
   更新汉语解释_文本条=new TextField(16);
   填加英文单词_文本条=new TextField(16);
   填加汉语解释_文本条=new TextField(16);
   删除英文单词_文本条=new TextField(16);
   删除汉语解释_文本条=new TextField(16);
   查询按钮=new Button("查询");
   更新按钮=new Button("更新");
   填加按钮=new Button("填加");
   删除按钮=new Button("删除");
   清空按钮=new Button("  清空文本框  ");
   Panel p1=new Panel(),p2=new Panel(),p3=new Panel(),p4=new Panel();
   p1.add(new Label("输入要查询的英语单词:"));
   p1.add(  待查英文单词_文本条);
   p1.add(new Label("显示该单词的汉语解释:"));
   p1.add(汉语解释_文本条);
   p1.add(查询按钮);
   p3.add(new Label("输入要添加的英语单词:"));p3.add(    填加英文单词_文本条);
   p3.add(new Label("输入该单词的汉语解释:"));p3.add(填加汉语解释_文本条);
   p3.add(填加按钮);
   p2.add(new Label("输入要更新的英语单词:"));p2.add(  更新英文单词_文本条);
   p2.add(new Label("输入该单词更新的汉语解释:"));
   p2.add(更新汉语解释_文本条);
```

```
    p2.add(更新按钮);
    p4.add(new Label("输入要删除的英语单词:"));p4.add(删除英文单词_文本条);
    p4.add(删除按钮);
    p4.add(new Label("                                    "));
    p4.add(清空按钮);

    add(p1);add(p3);add(p2);add(p4);
    查询按钮.addActionListener(this);
    更新按钮.addActionListener(this);
    填加按钮.addActionListener(this);
    删除按钮.addActionListener(this);
    清空按钮.addActionListener(this);
    addWindowListener(new WindowAdapter()
  {public void windowClosing(WindowEvent e)
  {setVisible(false);System.exit(0);   } } );
  }
  public void actionPerformed(ActionEvent e)
{if(e.getSource()==查询按钮)
  {查询记录=0;
  try{ 查询();}
  catch(SQLException ee) {}
  }
else if(e.getSource()==更新按钮)
{ try{ 更新();}
catch(SQLException ee) {}
}
else if(e.getSource()==填加按钮)
{try{ 填加();}
catch(SQLException ee) {}
}
else if(e.getSource()==删除按钮)
{try{ 删除();}
catch(SQLException ee) {}
}
  else if(e.getSource()==清空按钮)
{ 待查英文单词_文本条.setText("");
汉语解释_文本条.setText("");
填加英文单词_文本条.setText("");
填加汉语解释_文本条.setText("");
更新英文单词_文本条.setText("");
更新汉语解释_文本条.setText("");
删除英文单词_文本条.setText("");
}
}
```

```
public void 查询() throws SQLException
{ String cname,ename;
Con=DriverManager.getConnection("jdbc:odbc:test","gxy","ookk");
ResultSet rs=Stmt.executeQuery("SELECT * FROM cidian ");
while (rs.next())
{ ename=rs.getString("单词"); cname=rs.getString("解释");
if(ename.equals( 待查英文单词_文本条.getText().trim()))
{ 汉语解释_文本条.setText(cname);查询记录=1; break; }
}
Con.close();
if(查询记录==0)
{汉语解释_文本条.setText("没有该单词"); }
}
public void 更新() throws SQLException
{   String s1=""+更新英文单词_文本条.getText().trim()+"",
s2=""+更新汉语解释_文本条.getText().trim()+"";
String temp="UPDATE cidian SET 解释 ="+s2+" WHERE 单词 ="+s1 ;
Con=DriverManager.getConnection("jdbc:odbc:test","gxy","ookk");
Stmt.executeUpdate(temp);  Con.close();
JOptionPane.showMessageDialog(this,"更新成功！");
}
public void 填加() throws SQLException
{ String s1=""+填加英文单词_文本条.getText().trim()+"",
s2=""+填加汉语解释_文本条.getText().trim()+"";
String temp="INSERT INTO cidian VALUES ("+s1+","+s2+")";
Con=DriverManager.getConnection("jdbc:odbc:test","gxy","ookk");
Stmt.executeUpdate(temp);
Con.close();
JOptionPane.showMessageDialog(this,"填加成功！");
}
public void 删除() throws SQLException
{ String s1=""+删除英文单词_文本条.getText().trim()+"";
String temp="DELETE FROM cidian WHERE 单词 ="+s1 ;
Con=DriverManager.getConnection("jdbc:odbc:test","gxy","ookk");
Stmt.executeUpdate(temp);  Con.close();
JOptionPane.showMessageDialog(this,"删除成功！");
}
}
public class Database
{ public static void main(String args[])
{DataWindow window=new DataWindow();window.pack();
}
}
```

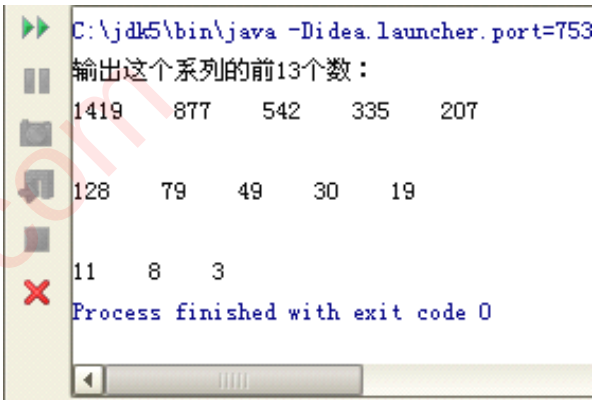
第 11 章 常见数据结构及算法分析

【1】设有一数列：a_1=3，a_2=8，……，an=2a_{n-1}+2a_{n-2}，使用堆栈结构输出 a_n 的若干项。

[解答]：代码如下，运行程序时需要输入一个参数，指出想要输出数列的前多少项

```
import java.util.Stack;
public class StackShow {
    public static void main(String[] args) {
        Stack st = new Stack();
        int count = Integer.valueOf(args[0]).intValue();
        int temp;
        Integer first = new Integer(3);
        Integer second = new Integer(8);
        st.add(first);
        st.add(second);
        for (int i = 0; i < count - 2; i++) {
            temp = first.intValue() + second.intValue();
            st.add(new Integer(temp));
            first = second;
            second = new Integer(temp);
        }
        System.out.println("输出这个系列的前" + count + "个数：");
        Object result[] = st.toArray();
        int wanghang = 0;
        for (int i = result.length - 1; i >= 0 ; i--) {
            System.out.print(st.pop() + "      ");
            wanghang++;
            if(wanghang % 5 == 0){
                System.out.println("\n");
            }
        }
    }
}
```

输入 13 时的运行结果如下：



【2】编写一程序，用哈希表实现学生成绩单的存储与查询。

[解答]：学生类 Student，代码如下：

```
class Student{
    private String no;
    private String name;
    private Integer score;
    public String getNo() {
        return no;
    }
    public void setNo(String no) {
        this.no = no;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getScore() {
        return score;
    }
    public void setScore(Integer score) {
        this.score = score;
    }
    public String toString(){
        return "学号：" + no + "    姓名：" + name + "    成绩：" + score;
    }
}
```

主类 HashTest，代码如下：

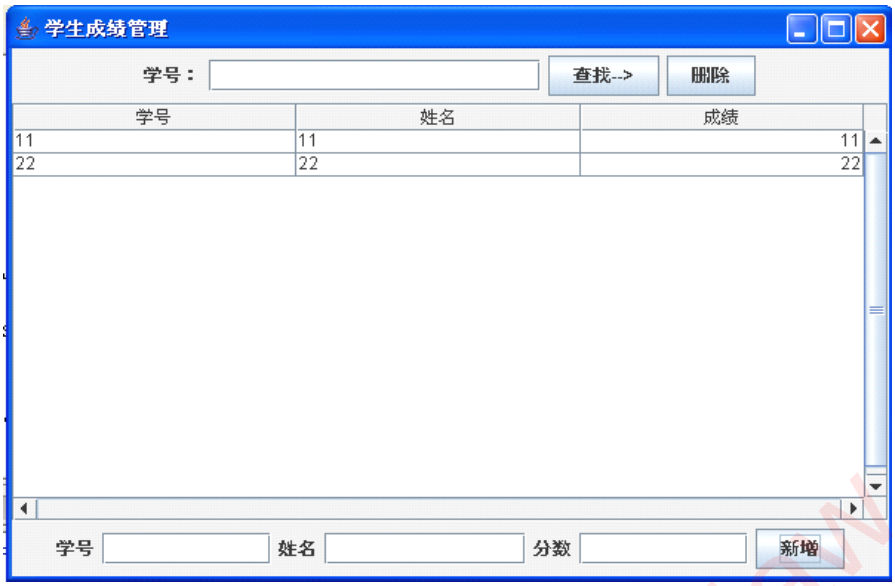
```
import javax.swing.*;
import java.util.Vector;
import java.util.Hashtable;
import java.awt.*;
```

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
public class HashTest extends JFrame {
    JLabel lblsearchbyidorname;
    JTextField txfidorname;
    JButton btnsearchbyidorname;
    JTable reader;
    JButton btnadd;
    JButton btndelete;
    Hashtable ht;
    Vector colnames;
    JLabel lblno;
    JLabel lblname;
    JLabel lblscore;
    JTextField addno;
    JTextField addname;
    JTextField addscore;
    Vector data;
    public HashTest() throws HeadlessException {
        super("学生成绩管理");
        ht = new Hashtable();
        lblsearchbyidorname = new JLabel("学号：");
        txfidorname = new JTextField(20);
        lblno = new JLabel("学号");
        lblname = new JLabel("姓名");
        lblscore = new JLabel("分数");
        addno = new JTextField(10);
        addname = new JTextField(12);
        addscore = new JTextField(10);
        btnsearchbyidorname = new JButton("查找-->");
        btnadd = new JButton("新增");
        btndelete = new JButton("删除");
        colnames = new Vector();
        colnames.add("学号");
        colnames.add("姓名");
        colnames.add("成绩");
        data = new Vector();
        reader = new JTable(new ReaderTableModel(data,colnames));
        reader.setPreferredSize(new Dimension(700,260));
        JPanel pnlsearch = new JPanel();
        pnlsearch.add(lblsearchbyidorname);
        pnlsearch.add(txfidorname);
        pnlsearch.add(btnsearchbyidorname);
        pnlsearch.add(btndelete);
```

```
        JScrollPane scptable = new JScrollPane(reader,
                ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
                ScrollPaneConstants.HORIZONTAL_SCROLLBAR_ALWAYS);
        JPanel pnladd = new JPanel();
        pnladd.add(lblno);
        pnladd.add(addno);
        pnladd.add(lblname);
        pnladd.add(addname);
        pnladd.add(lblscore);
        pnladd.add(addscore);
        pnladd.add(btnadd);
        reader.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        ScoreHandler sh = new ScoreHandler();
        btnadd.addActionListener(sh);
        btndelete.addActionListener(sh);
        btnsearchbyidorname.addActionListener(sh);
        Container c = getContentPane();
        c.add(pnlsearch,BorderLayout.NORTH);
        c.add(scptable,BorderLayout.CENTER);
        c.add(pnladd,BorderLayout.SOUTH);
        setSize(600,400);
        setVisible(true);
    }
    public static void main(String[] args) {
        new HashTest();
    }
    class ScoreHandler implements ActionListener{
        public void actionPerformed(ActionEvent e) {
            JButton btn = (JButton)e.getSource();
            if(btn == btnsearchbyidorname){
                Object obj = ht.get(txfidorname.getText().trim());
                if(obj == null){
                    JOptionPane.showMessageDialog(null,"没有找到！");
                }else{
                  JOptionPane.showMessageDialog(null,"查询结果如下：\n" + obj.toString());
                }
            }else if(btn == btnadd){
                Student stu = new Student();
                stu.setName(addname.getText().trim());
                stu.setNo(addno.getText().trim());
                stu.setScore(Integer.valueOf(addscore.getText().trim()));
                ht.put(stu.getNo(),stu);
                addDataToTable(stu);
                addname.setText("");
```

```
                addno.setText("");
                addscore.setText("");
            }else if(btn == btndelete){
                int index = reader.getSelectedRow();
                if (index == -1){
                    JOptionPane.showMessageDialog(null,"你没有选择学生！ ");
                }else{
                    String no = (String)reader.getValueAt(index,0);
                    Student stu = (Student)ht.remove(no);
                 JOptionPane.showMessageDialog(null,"学生成绩删除！\n" + stu.toString());
                    data.remove(index);
                    reader.repaint();
                }
            }
        }
    }
    public void addDataToTable(Student stu){
        Vector temp = new Vector();
        temp.add(stu.getNo());
        temp.add(stu.getName());
        temp.add(stu.getScore());
        data.add(temp);
        reader.repaint();
    }
}
```

运行效果如下：



查找：

【3】（走迷宫）下列由符号＃和点组成的图形代表一个迷宫，用一个双下标数组存放。其中，＃代表迷宫的墙，点代表路径，只有数组中含有点的地方才能走。

[解答]：对于用如下二维数组表示的迷宫

```
{{1,1,0,1,1,1,1,1},
{1,0,0,1,0,0,0,1},
{1,1,0,0,0,1,1,1},
{1,0,0,1,0,0,0,1},
{1,1,1,1,0,1,1,1},
{1,0,0,0,0,0,0,1},
{1,0,1,0,1,0,0,1},
{1,1,1,0,1,1,1,1}}
```

为了简化算法，假设入口是（0，2），出口是（7，3），程序如下：

类 Position 用来描述点在数组中的位置和该点旁边四个点是否可以通过的状态，程序如下：

```
class Position{
    private int px;
    private int py;
    private boolean forwardFlag;
    private boolean backFlag;
    private boolean leftFlag;
    private boolean rightFlag;

    public Position(int px,int py,int[][] allData){
        this.px = px;
        this.py = py;
        int row = allData.length;
        int col = allData[0].length;
        int forward = px + 1;
        int back = px - 1;
        int left = py + 1;
        int right = py - 1;
```

```
        if(forward == row || allData[forward][py] == 1){
            forwardFlag = false;
        }else{
            forwardFlag = true;
        }
        if(back == -1 || allData[back][py] == 1){
            backFlag = false;
        }else{
            backFlag = true;
        }
        if(left == col || allData[px][left] == 1){
            leftFlag = false;
        }else{
            leftFlag = true;
        }
        if(right == -1 || allData[px][right] == 1){
            rightFlag = false;
        }else{
            rightFlag = true;
        }
    }

    public boolean isForwardFlag() {
        return forwardFlag;
    }

    public void setForwardFlag(boolean forwardFlag) {
        this.forwardFlag = forwardFlag;
    }

    public boolean isBackFlag() {
        return backFlag;
    }

    public void setBackFlag(boolean backFlag) {
        this.backFlag = backFlag;
    }

    public boolean isLeftFlag() {
        return leftFlag;
    }

    public void setLeftFlag(boolean leftFlag) {
        this.leftFlag = leftFlag;
```

```
    }

    public boolean isRightFlag() {
        return rightFlag;
    }

    public void setRightFlag(boolean rightFlag) {
        this.rightFlag = rightFlag;
    }

    public int getPx() {
        return px;
    }
    public void setPx(int px) {
        this.px = px;
    }
    public int getPy() {
        return py;
    }
    public void setPy(int py) {
        this.py = py;
    }
    public String toString(){
        return "(" + px +    "," + py + ")";
    }
}
```

主类：

```
import java.util.Stack;
import java.util.Enumeration;
import java.util.Iterator;
public class PassMaze {
    int[][] maze = {{1,1,0,1,1,1,1,1},
                    {1,0,0,1,0,0,0,1},
                    {1,1,0,0,0,1,1,1},
                    {1,0,0,1,0,0,0,1},
                    {1,1,1,1,0,1,1,1},
                    {1,0,0,0,0,0,0,1},
                    {1,0,1,0,1,0,0,1},
                    {1,1,1,0,1,1,1,1}};
    private Stack st, direct;
    public PassMaze() {
        direct = new Stack();
        st = new Stack();
        Position current = new Position(0,2,maze);
```

```
        st.push(current);
        while(!isSuccessful(current)){
            boolean status = calPassWay(current);
            if(!status){
                st.pop();
                if(st.empty()){
                    break;
                }
                setFlag((Position)st.peek(),(String)direct.pop());
            }
            current = (Position)st.peek();
        }
        if(st.empty()){
            System.out.println("迷宫没有出路！");
        }else{
            System.out.println("迷宫的出路如下：");
            Iterator it = st.iterator();
            while(it.hasNext()){
                System.out.println(it.next().toString());
            }
        }
    }
    private boolean calPassWay(Position pos){
        boolean result = false;
        if(pos.isBackFlag()){
            Position temp = new Position(pos.getPx() - 1,pos.getPy(),maze);
            temp.setForwardFlag(false);
            if(contains(temp)){
                pos.setBackFlag(false);
            }else{
                st.push(temp);
                direct.push("Back");
                result = true;
            }
        }else if(pos.isForwardFlag()){
            Position temp = new Position(pos.getPx() + 1,pos.getPy(),maze);
            temp.setBackFlag(false);
            if(contains(temp)){
                pos.setForwardFlag(false);
            }else{
                st.push(temp);
                direct.push("Forward");
                result = true;
            }
```

```
        }else if(pos.isLeftFlag()){
            Position temp = new Position(pos.getPx(),pos.getPy() + 1,maze);
            temp.setRightFlag(false);
            if(contains(temp)){
                pos.setLeftFlag(false);
            }else{
                st.push(temp);
                direct.push("Left");
                result = true;
            }
        }else if(pos.isRightFlag()){
            Position temp = new Position(pos.getPx(),pos.getPy() - 1,maze);
            temp.setLeftFlag(false);
            if(contains(temp)){
                pos.setRightFlag(false);
            }else{
                st.push(temp);
                direct.push("Right");
                result = true;
            }
        }
        return result;
    }
    private boolean contains(Position p){
        boolean result = false;
        Iterator it = st.iterator();
        while(it.hasNext()){
            Position temp = (Position)it.next();
            if(p.getPx() == temp.getPx() && p.getPy() == temp.getPy()){
                result = true;
                break;
            }
        }
        return result;
    }
    private void setFlag(Position current,String dirction){
        if("Back".equals(dirction)){
            current.setBackFlag(false);
        }else if("Forward".equals(dirction)){
            current.setForwardFlag(false);
        }else if("Left".equals(dirction)){
            current.setLeftFlag(false);
        }else if("Right".equals(dirction)){
            current.setRightFlag(false);
```

```
            }
        }
        private boolean isSuccessful(Position pt){
            return pt.getPx() == 7 && pt.getPy() == 3;
        }
        public static void main(String[] args) {
            new PassMaze();
        }
}
```

第 12 章 J2ME 程序设计基础

【1】安装并配置 J2ME 的运行环境。

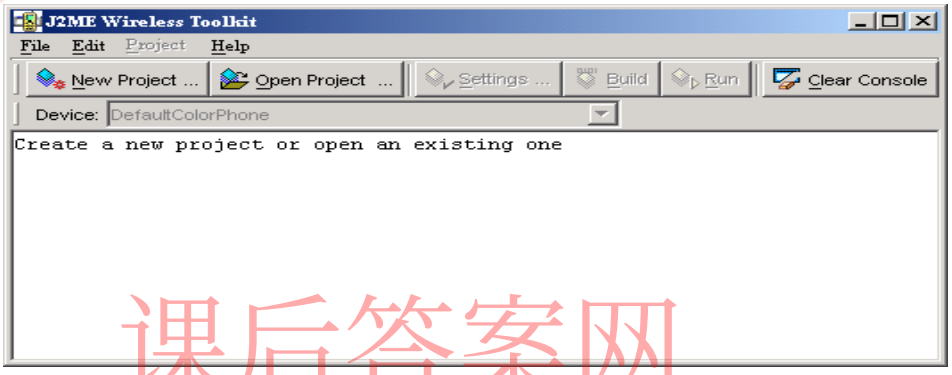
[解答]：安装步骤如下：

1、 安装 JDK1.4；

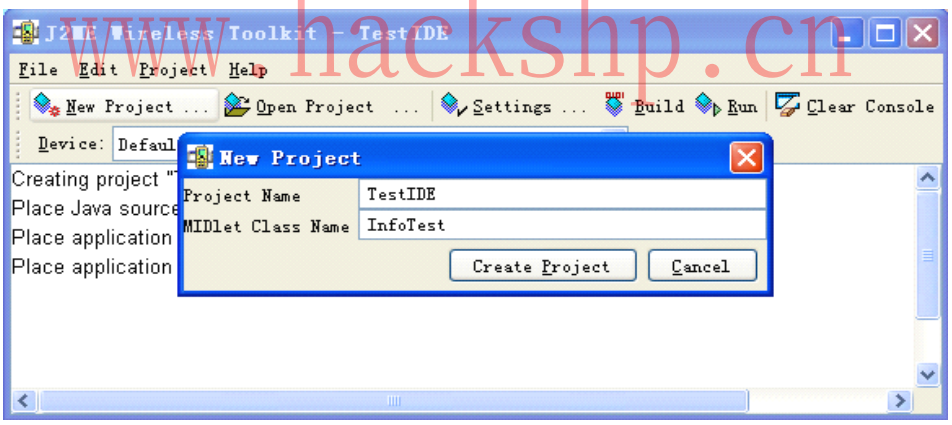
2、 安装 J2ME Wireless Toolkit2.2(WTK22)；

测试运行环境：使用 WTK 创建第一个 MIDlet 程序，步骤如下：

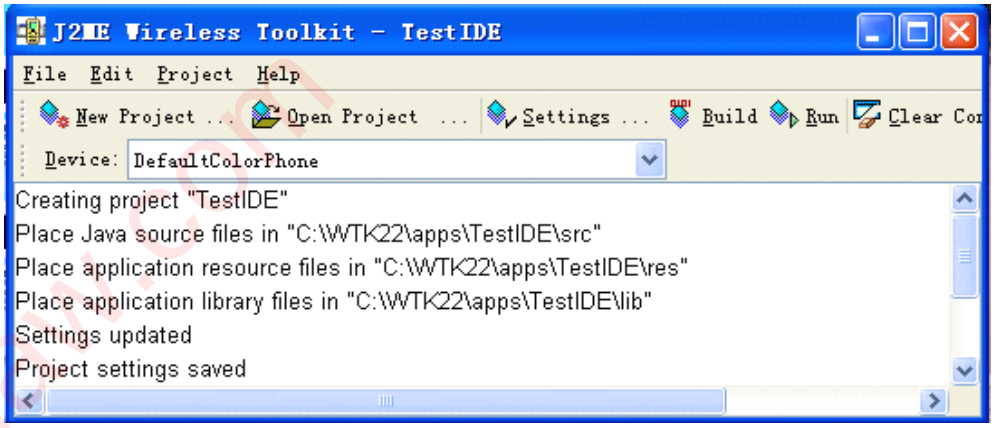
打开开发环境：开始 > 程序 > J2ME Wireless Toolkit 2.2 > KToolbar，如下：



（2）新建项目。项目名称：合法的变量名称就行；MIDlet 类名称：该类为程序运行入口类，如下所示：



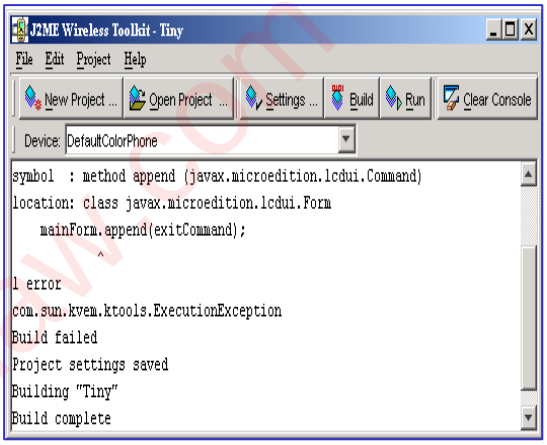
（3）编写程序。使用任意的 IDE，编写类 InfoTest.java（MIDlet 类名称，与新建项目时的名称相同），同时必须保存在 Java 源文件目录下。如果有图片等资源文件，必须保存在应用程序源文件目录下。如下所示：



源程序如下：

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class InfoTest extends MIDlet{
 private Display display;
 public InfoTest(){
       display=Display.getDisplay(this);
 }
     protected void startApp(){
      Alert alert =new Alert("手机信息测试");
           alert.setTimeout(Alert.FOREVER);
           String icon="/zsm.jpg"; //从资源目录开始寻找
           try{
            Image image=Image.createImage(icon);
            alert.setImage(image);
           }catch(java.io.IOException x){System.out.println("出错了");}
           display.setCurrent(alert);
     }
 protected void pauseApp(){}
 protected void destroyApp(boolean unconditional){}
}
```

（4）编译与运行



【2】设计一个公告显示程序。

[解答]：代码如下：

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.TextBox;
import javax.microedition.lcdui.TextField;
import javax.microedition.lcdui.Ticker;
public class ShowInfo extends MIDlet {
    private Display dp;
    private TextBox tb;
    public ShowInfo(){
        super();
        tb = new TextBox("信息公告","",200, TextField.ANY);
    }
    protected void startApp() throws MIDletStateChangeException {
        dp = Display.getDisplay(this);
        tb.setTicker(new Ticker("今天全校停课，哈哈！"));
        tb.setString("由于天气原因，全校停课一个月，回家种地去！");
        dp.setCurrent(tb);
    }
    protected void pauseApp() {
    }
    protected void destroyApp(boolean b) throws MIDletStateChangeException {
    }
}
```

运行结果如下：

【3】应用记录管理系统 RMS 建立一个同学通讯录。

[解答]：同学通讯录的要求如下：

1、 每个联系人包括两项信息：姓名，电话号码；

2、 能对联系人进行增加、删除、修改和查询操作；

3、 通过一个 MIDlet 程序实现与用户的交互（实现界面）。

程序如下：

TelBean.java：描述联系人实体，既是用于传值的 JavaBean，又提供该类对象与字节数组相互转换的方法，代码如下：

```
import java.io.*;
public class TelBook {
    private String name;
    private String tel;
    public String getName() {
        return name;
    }
     public void setName(String name) {
        this.name = name;
    }
    public String getTel() {
        return tel;
    }
    public void setTel(String tel) {
        this.tel = tel;
    }
    public byte[] toByteArray(){
        byte[] data = null;
        ByteArrayOutputStream bout = new ByteArrayOutputStream();
```

```
            DataOutputStream dout = new DataOutputStream(bout);
            try {
                dout.writeUTF(this.name);
                dout.writeUTF(this.tel);
                data = bout.toByteArray();
                dout.close();
                bout.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return data;
        }
        public void initTelBean(byte[] rec){
            ByteArrayInputStream bin = new ByteArrayInputStream(rec);
            DataInputStream din = new DataInputStream(bin);
            try {
                this.name = din.readUTF();
                this.tel = din.readUTF();
                din.close();
                bin.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

TelModel.java：操作 RecordStore，实现对电话本的增、删、查、改功能，代码如下：

```
import javax.microedition.rms.RecordStore;
import javax.microedition.rms.RecordStoreException;
import javax.microedition.rms.RecordStoreNotOpenException;
import java.io.*;
public class TelModel {
    private RecordStore rs = null;
    public TelModel() {
        try {
            rs = RecordStore.openRecordStore("Pref",true);
        } catch (RecordStoreException e) {
            e.printStackTrace();
        }
    }
    public int addRecord(TelBook mn){
        int re = -1;
        try {
            re = rs.addRecord(mn.toByteArray(),0,mn.toByteArray().length);
        } catch (RecordStoreException e) {
```

```
            e.printStackTrace();
            return -1;
        }
        return re;
    }
    public TelBook getRecord(int recordID){
        TelBook note = new TelBook();
        try {
            byte[] MyNoteBytes = rs.getRecord(recordID);
            note.initTelBean(MyNoteBytes);
        } catch (RecordStoreException e) {
            e.printStackTrace();
        }
        return note;
    }
    public boolean setRecord(int recordID,TelBook mn){
        try {
            byte[] temp = mn.toByteArray();
            rs.setRecord(recordID,temp,0,temp.length);
        } catch (RecordStoreException e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }
    public boolean deleteRecord(int recordID){
        try {
            rs.deleteRecord(recordID);
        } catch (RecordStoreException e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }
    public int getRecordCounts(){
        try {
            return rs.getNumRecords();
        } catch (RecordStoreNotOpenException e) {
            e.printStackTrace();
        }
        return -1;
    }
    public void Close(){
        try {
```

```
                rs.closeRecordStore();
            } catch (RecordStoreException e) {
                e.printStackTrace();
            }
        }
}
```

TelView.java：用户界面，得到用户的输入并显示系统输出，代码如下：

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;
import javax.microedition.lcdui.*;

public class RecordMidlet extends MIDlet implements CommandListener {
    private Form mForm;
    private Form note;
    private Display dp;
    private TelModel rs;
    private TextField mSubject,mContent,mID;
    private int currentID;
    private int status = -1;
    private Command CMD_ADD;
    private Command CMD_DEL;
    private Command CMD_EDIT;
    private Command CMD_EXIT;
    private Command CMD_INFO;
    private Command CMD_OK;
    private Command CMD_CANCEL;
    public RecordMidlet(){
        rs = new TelModel();
        note = new Form("Store Book");
        mSubject = new TextField("姓名","",20,0);
        mContent = new TextField("号码","",20,0);
        mID = new TextField("记录号","",10,TextField.NUMERIC);
        CMD_ADD = new Command("Add Record",Command.ITEM,1);
        CMD_DEL = new Command("Delete Record",Command.ITEM,1);
        CMD_EDIT = new Command("Edit Record",Command.ITEM,1);
        CMD_EXIT = new Command("Exit",Command.EXIT,1);
        CMD_INFO = new Command("Recode Store Information",Command.EXIT,1);
        CMD_OK = new Command("Ok",Command.OK,1);
        CMD_CANCEL = new Command("Cancel",Command.CANCEL,1);
        note.addCommand(CMD_CANCEL);
        note.addCommand(CMD_OK);
        note.setCommandListener(this);
    }
```

```
protected void startApp() throws MIDletStateChangeException {
    dp = Display.getDisplay(this);
    if(mForm == null){
        mForm = new Form("电话本");
        mForm.addCommand(CMD_ADD);
        mForm.addCommand(CMD_DEL);
        mForm.addCommand(CMD_EDIT);
        mForm.addCommand(CMD_EXIT);
        mForm.addCommand(CMD_INFO);
        mForm.setCommandListener(this);
    }
    dp.setCurrent(mForm);
}
protected void pauseApp() {
}
protected void destroyApp(boolean b) throws MIDletStateChangeException {
    rs.Close();
}
public void commandAction(Command command, Displayable displayable) {
    if(command == CMD_EXIT){
        notifyDestroyed();
    }else if(command == CMD_CANCEL){
        dp.setCurrent(mForm);
    }else if(command == CMD_ADD){
        note.deleteAll();
        note.append(mSubject);
        mSubject.setString("");
        note.append(mContent);
        mContent.setString("");
        status = 0;
        dp.setCurrent(note);
    }else if(command == CMD_EDIT){
        note.deleteAll();
        note.append(mID);
        status = 1;
        dp.setCurrent(note);
    }else if(command == CMD_DEL){
        note.deleteAll();
        note.append(mID);
        status = 3;
        dp.setCurrent(note);
    }
    if(command == CMD_OK){
        Alert a = new Alert("");
```

```
        if(status == 0){
            TelBook my = new TelBook();
            my.setTel(mContent.getString());
            my.setName(mSubject.getString());
            int id = rs.addRecord(my);
            a.setString("Add record successfully!" + id);
            a.setTimeout(3000);
            dp.setCurrent(a,mForm);
        }else if(status == 1){
            currentID = Integer.parseInt(mID.getString());
            TelBook temp = rs.getRecord(currentID);
            mSubject.setString(temp.getName());
            mContent.setString(temp.getTel());
            note.deleteAll();
            note.append(mSubject);
            note.append(mContent);
            dp.setCurrent(note);
            status = 2;
            return;
        }else if(status == 2){
            TelBook temp = new TelBook();
            temp.setName(mSubject.getString());
            temp.setTel(mContent.getString());
            boolean re = rs.setRecord(currentID,temp);
            if(re){
                a.setString("Edit record successfully");
            }else{
                a.setString("Eidt record failure");
            }
            dp.setCurrent(a,mForm);
            status = -1;
        }else if(status == 3){
            currentID = Integer.parseInt(mID.getString());
            boolean re = rs.deleteRecord(currentID);
            if(re){
                a.setString("Delete record successfully");
            }else{
                a.setString("Delete recode failure");
            }
            dp.setCurrent(a,mForm);
        }
    }
    if(command == CMD_INFO){
        String temp = "";
```

```
                temp += "The current RecordStore have" + rs.getRecordCounts() + "records";
                StringItem ss = new StringItem("RecordStore information\n",temp);
                note.deleteAll();
                note.append(ss);
                dp.setCurrent(note);
                status = 4;
            }
        }
    }
```

运行结果如下：

主界面：

新增界面：

编辑界面：