Java 程序设计 （清华大学出版社　　林巧民主编）

第 1 章

1、结合实际生活实践，简述计算机的用处？

（1）科学计算

（2）信息管理

（3）计算机辅助工程

（4）自动控制

（5）数据处理

2、按照性能指针分类的话，都有哪些类型的计算机。

巨型计算机

大/中型计算机

小型计算机

微型计算机

工作站

服务器

3、计算机系统由哪两部分组成的？计算机硬件结构由哪几部分组成？它们各自有什么作用？

一台计算机由硬件和软件组成。　　一台典型的计算机由五大部分组成。　　这五大部分是： 运算器，控制器，存储器，输入设备和输出设备。

运算器是执行算术运算和逻辑运算的部件。

控制器是向计算机其他部分发送命令的部件。

存储器是计算机用来存储数据、信息的部件。

输入设备就是外界向计算机输入信息设备。

输出设备恰好与输入设备的作用相反，它将处理过后的信息输出呈现给用户。

4、简述光栅扫描显示器的组成及工作原理？

光栅扫描显示器由主要由显示存储器、图像生成器、彩色表、　　CRT 控制器和 CRT 监视器 5 个部分组成。 首先由图形生成器根据主机发送来的画图命令转换成相应点阵存入到显示存储器中，即在显示存储器中生成所显示画面位图。然后，　　CRT 控制器一方面产生水平和垂直同步信号并将其送到监视器，使　　CRT 电子束不断地自上而下、自左向右进行扫描，形成光栅； 另一方面， 又根据电子束在屏幕上的行列位置，　　不断地读出显示存储器中对应位置的像素值。此时彩色表中对应值控制　　R、 G、 B 电子束，在屏幕对应点生成需要的像素值。为了使显示画面不产生闪烁，上述过程要反复进行，一般要求　　CRT 的帧频为 60 帧/秒以上。

5、什么是计算机软件？列举你所知道的一些常用软件？

软件实际上就是指我们计算机上所有可以运行的程序、 代码、 以及文档和数据的总和。 大到操作系统小到一个文本编辑器，都属于软件的范畴。

常用软件：操作系统， Office 办公软件， QQ，游戏等。

6、计算机病毒是什么东西？如何有效防止病毒？

计算机病毒是一种人为制造的、在计算机运行中对计算机信息或系统起破坏作用的程序。这种程序不是独立存在的， 它隐蔽在其他可执行的程序之中， 既有破坏性，又有传染性和潜伏性。 轻则影响机器运行速度， 使机器不能正常运行；重则使机器处于瘫痪， 会给用户带来不可估量的损失。通常就把这种具有破坏作用的程序称为计算机病毒。

预防措施：

（1）思想上要重视，使用移动存储设备如 u 盘拷贝东西要小心病毒，上网时尽量只访问比较正规的网站，从源头上遏制病毒的传播。

（2）注意及时更新系统，打全补丁程序，有效阻止病毒的入侵。

（3）建议安装杀毒防毒软件，以及网络防火墙等。

（4）提高自身的计算机水平，以有效应付病毒。

7、名词解释：指令，机器语言，汇编语言，高级语言。

指令由操作码和操作数组成：

? 操作码 要完成的操作类型或性质

? 操作数 操作的内容或所在的地址

机器语言 是由 0 和 1 二进制代码按一定规则组成的、能被机器直接理解和执行的指令集合。

将机器指令的代码用英文助记符来表示，代替机器语言中的指令和数据。例如用 ADD 表示加、 SUB 表示减、 JMP 表示程序跳转等等，这种指令助记符的语言就是 汇编语言 。

高级语言 是用近似自然语言并按照一定的语法规则来编写程序的语言。 高级语言使程序员可以完全不用与计算机的硬件打交道，可以不必了解机器的指令系统，编程效率高。

8、将以下二进制数转换为对应的十进制数

（**1**）**111** （**2**）**10011** （**3**） （**4**）

（1）7 （2）19 （3）51 （4）140

9、将以下十进制数转换为对应的二进制数

（**1**）**32** （**2**）**97** （**3**）**256** （**4**）**500**

（1） （2） （3） （4）

10、简述什么是原码、反码和补码？

原码、反码和补码用来表示计算机中的数值信息：

原码：符号位 +数值大小

反码：正数反码同原码，负数的反码为除符号位外其他位都取反。

补码：正数同原码，负数的补码为在反码的最低位加“ 1 ”。

11、简述操作系统的功能以及你所知道的操作系统。

操作系统是对计算机硬件系统的一次扩充。 用户通过操作系统来使用计算机系统。 从而，使得用户能够方便、可靠、安全、高效地操纵计算机硬件和运行自己的程序。

DOS 操作系统

Windows 系列操作系统

UNIX 操作系统

Linux 操作系统

Solaris 操作系统

12、简述字符和汉字在计算机内的表示方法。

目前国际上使用最广泛的字符编码为 ASCII 码。

ASCII 码：是美国国家标准信息交换码（ American Standard Code for Information Interchange）的简称，一个字节的编码对应一个字符，最高位一般为 0，是 7 位编码，可表示 128 个不同字符，如为 1、为 E 等。

1981 年， GB2312-80 国家标准，其中有 6763 个汉字和 682 个非汉字字符，其字符及编码称为国标码又叫国际交换码。 GB2312 字符集的构成：

一级常用汉字 3755 个，按汉语拼音排列

二级常用汉字 3008 个，按偏旁部首排列

非汉字字符 682 个

汉字以双字节表示。 在计算机的汉字信息处理系统中， 处理汉字时要进行如下的代码转换：输入码→交换码→内部码→字形码。

第 2 章

1. Java 语言有哪些主要特点。

平台独立性

安全性

多线程

网络化

面向对象

2. 目前美国 **Sun** 公司提供的适用不同开发规模的 **JDK** 有哪些。

目前 Sun 共提供了三种不同的版本：微平台版 J2ME（Java 2 Platform Micro Edition ），标准版 J2SE（Java 2 Platform Standard Edition ）和企业版 J2EE（Java 2 Platform Enterprise Edition ），这三种版本分别适用于不同的开发规模和类型， 对于普通 Java 开发人员和一般学习者来说，选用标准版 J2SE 就可以了，学会了 J2SE，再学 J2ME 或 J2EE 就比较容易上手，因为它们之间虽有所侧重，但相似之处很多，尤其是语言本身是一样的，都是 Java。

3. Java Application 的开发步骤有哪些。

Java Application 的开发步骤：

（1）下载 JDK 软件并安装；

（2）配置相应的环境变量（ path 和 classpath）；

（3）编写 Java 源程序（文本编辑器或集成开发环境 IDE）；

（4）编译 Java 源程序，得到字节码文件（ javac *.java）；

（5）执行字节码文件（ java 字节码文件名）。

4. 什么是环境变量，设置环境变量的主要目的是什么。

环境变量的配置主要是为了进行“寻径” ，也即让程序能找到它需要的文件，所以设置的内容就是一些路径。

5. 不参考书本， 试着编写一个简单的 **Java Application** 程序，实现在 **Dos** 窗口输出“**Welcome to Nanjing City!** ”字符串。并对该程序进行编译和运行。

```
public class Hello {
    public static void main(String args[])
    {
        System.out.println(" Welcome to Nanjing City! ");
    }
}
```

6. 编写一个 **Java Application** 程序，实现分行显示字符串“ **Welcome to Nanjing City** ”中的四个单词。

```
public class Hello {
    public static void main(String args[])
    {
        System.out.println(" Welcome ");
        System.out.println(" to ");
        System.out.println(" Nanjing ");
        System.out.println(" City! ");
    }
}
```

第 3 章

1. **Java** 语言对于合法标识符的规定是什么？指出以下哪些为合法标识符。

a a2 3a *a _a $a int a%

在 Java 语言中，标识符必须以字母、美元符号或者下划线打头，后接字母、数字、下划线或美元符号串。另外， Java 语言对标识符的有效字符个数不做限定。

合法的标识符：

a a2 _a $a

2. 变量的涵义是什么？变量名与变量值的关系？

在程序执行过程中其值可以改变的数据，称为变量，它本质上代表了内存中的一小块空间。每个变量都必须有对应的名称来标识它，即变量名，而变量空间所存放的数据则称为变量值。

3. **Java** 语言提供哪些基本的数据类型，为什么要提供这些不同的数据类型？

布尔型： boolean

整型： byte、short、int 以及 long

浮点型（实型）：float 以及 double

字符型： char

数据类型代表了数据的存储格式和处理方式，虽然严格来说计算机只能识别“0”和“1”，但是，有了数据类型以后，计算机的识别能力就被人为扩展了，它能够识别整数、实数以及字符等。

4. 赋值语句的涵义是什么 **?**

数据传递。

5. 数据类型强制转换的原则是什么？如何转换？

对于变窄转换，如 long 到 short、double 到 float，或者不兼容转换： float 到 short、char 到 short 等，则需要进行强制转换。

```
float f = 11.5;
short b ;
b = （short）f; （强制转换）
```

6. 每一条程序语句都应以分号来结束，这个分号能否用中文输入模式下输入的分号，为什么？

不能。首先中英文输入模式下输入的分号是不同的，不但外观不同，其存储所需的空间也是不同的，Java 编译器只能识别英文的分号，对于中文分号，将会报错。

第 4 章

1. 假定乘坐飞机时，每位顾客可以免费托运 20kg 以内的行李，超过部分按每公斤收费 1.2 元，以下是相应的计算收费程序。该程序存在错误，请找出。

```
public class Test
{
    public static void main(String[] args) throws IOException
    {
        float   w,fee;
         //以下代码为通过控制台交互输入行李重量
        InputStreamReader reader=new InputStreamReader(System.in);
        BufferedReader input=new BufferedReader(reader);
        System.out.println(" 请输入旅客的行李重量   :");
        String temp=input.readLine();
        w = Float.parseFloat(temp);    //字符串转换为单精度浮点型
        fee = 0;
```

```
            if ( w > 20);
                fee = (float)1.2 * (w-20);
            System.out.println(" 该旅客需交纳的托运费用：    "+fee+" 元 ");
        }
}
```

缺少 import java.io.*; 语句

2. 有一条长的阶梯，如果每步 **2** 阶，则最后剩 **1** 阶，每步 **3** 阶则剩 **2** 阶，每步 **5** 阶则剩 **4** 阶，每步 **6** 阶则剩 **5** 阶，只有每步 **7** 阶的最后才刚好走完，一阶不剩，问这条阶梯最少共有多少阶？找出以下求解程序的错误所在。

```
public class Test
{       public static void main(String[] args)
            {
                int i;
                while(i%2==1&&i%3==2&&i%5==4&&i%6==5&&i%7==0)
                {
                    i++;
                }
                System.out.println(" 这条阶梯最少有：   "+i+" 阶 ");
            }
}
```

1）变量 i 应进行初始化。

2） while 循环的条件表达式应再加上“ 非 ” ，这样才符合逻辑。

3. 试用单分支结构设计一判断用户输入值 **X**，当 **X** 大于零时求 **X** 值平方根，否则不执行任何操作的程序。

```
import java.io.*;
import java.lang.*;
public class Test
{       public static void main(String[] args) throws IOException
            {
                    float    x;
            InputStreamReader reader=new InputStreamReader(System.in);
            BufferedReader input=new BufferedReader(reader);
            System.out.println(" 请输入  x:");
            String temp=input.readLine();
            x = Float.parseFloat(temp);    //字符串转换为单精度浮点型
            if(x>0)
                    System.out.println("  平方根值   "+Math.sqrt(x));
            }
}
```

4. 从键盘读入两个字符，按照字母表排序顺序，将前面的字符置于 **A**，排后面的字符置于 **B**。请设计实现该程序。

```
import java.io.*;
public class Test
{       public static void main(String[] args) throws IOException
                {
                        char    A,B,c1,c2;
                        c1 = (char)System.in.read();;
                        c2 = (char)System.in.read();;
                        if(c1>c2)
                        {
                            A=c2;
                            B=c1;
                         }else
                        {
                                A=c1;
                            B=c2;
                        }
                       System.out.println("A="+A+" B="+B);
                }
}
```

5. 用穷举法求出 **3** 位数中百、十、个位数的立方和就是该数的数。

```
public class Test
{       public static void main(String[] args)
                {
                    int a,b,c,x=100;
                    while(x<1000){
                        a=x%10;
                        b=(x%100-a)/10;
                        c=(x-x%100)/100;
                        if(a*a*a+b*b*b+c*c*c==x)
                            System.out.println(x);
                        x+=1;
                     }
                }
}
```

6. 编程实现打印以下图案：

****

****

****

*****

***

*

```
public class Test
{       public static void main(String[] args)
        {
                int i,j,k; //      i 控制行数  , k 控制 * 的个数 ,j 控制空格数
                for(i=1;i<=6;i++)
                {
                    for(j=1;j<=i-1;j++)
                  System.out.print(" ");      //打印空格
                    for(k=1;k<=13-i*2;k++)
                    System.out.print("*");       //打印 *号
                System.out.println();       //换行
            }
        }
}
```

7. 统计 **1** 至 **1** 万共有多少个数是素数。

```
public class Test
{       public static void main(String[] args)
        {       int i,j,count=0;
                label:
                for(i=1;i<=10000;i++)           //查找 1 到 10000 以内的素数
                {       for(j=2;j<i;j++)                    //检验是否不满足素数条件
                            if (i%j==0)                     //不满足
                                    continue label;     //跳过后面不必要的检验
                        count++;       //计数
                 }
                 System.out.println(" 个数 :"+count);
         }
}
```

8. 打印输出斐波纳契数列的前 **12** 项。

斐波纳契数列的前 **12** 项如下：

第 **1** 项： **0**

第 **2** 项： **1**

第 **3** 项： **1**

第 **4** 项： **2**

第 **5** 项： **3**

第 **6** 项： **5**

第 **7** 项： **8**

第 **8** 项： **13**

第 **9** 项： **21**

第 **10** 项： **34**

第 **11** 项： **55**

第 **12** 项： **89**

```
public class Test
{       public static void main(String[] args)
        {    int i=0,j=1;
             for (int k=0;k<6;k++)
             {
                System.out.print(i+" "+j+" ");
                i=i+j;
                j=i+j;
             }
        }
}
```

9. 读程序，写结果。

```
import java.io.*;
public class Test
{
   public static void main(String[] args) throws IOException
  {
      char sex= 'f';
      switch (sex)
        {
          case  'm':    System.out.println("                    男性 ");
                    break;
          case  'f':    System.out.println("                    女性 ");
          case  'u':    System.out.println("                    未知 ");
        }
   }
}
```

女性
未知

10. 读程序，写结果。

```
public class Test
{       public static void main(String[] args)
        {
            int i ,s=0;
            for(i=1;i<=100;i++)
            {
```

```
            if(i%3==0)
                continue;
            s+=i;
        }
        System.out.println("s="+s);
    }
}
```

s=3367

11. 读程序，写结果。

```
public class Test
{    public static void main(String[] args)
    {
        int i ,s=0;
        for(i=1;i<=100;i++)
        {
            s+=i;
            if(s>100)
                break;
        }
        System.out.println("s="+s);
    }
}
```

s=105

12. 个位数是 **6**，且能被 **3** 整除的 **5** 位数有多少？

```
public class Test
{    public static void main(String[] args)
        {
            int i=10006,count=0;
            while(i<)
            {   if (i%3==0)
                    count++;
                i+=10;
            }
            System.out.println(" 符合条件的数共有   "+count+" 个");
        }
}
```

13. 用嵌套循环结构，设计一模拟电子钟的程序。

提示：定义三个变量分别代表“小时”、“分”和“秒”，根据电子钟分、秒、小时之间的关系，采用三重循环来控制各量的增加，并由输出语句将变化中的三个量分别予以输出显示，即为一模拟数字电子钟。 此外，**Java** 语言提供的延时方法为 **Thread.sleep(1000);1000** 的单位为毫秒，即延时 **1** 秒。

```
import java.util.*;
public class Test
{       public static void main(String[] args) throws InterruptedException
                {
                    int hour=12,min=0,sec=0;
                    while(hour<24)
                    {
                        while(min<60)
                    {
                        while(sec<60)
                        {    sec++;
                            Thread.sleep(1000);
                            System.out.println(hour+"  时 -"+min+"  分 -"+sec+" 秒 ");
                          }
                        sec=0;
                        min++;
                    }
                    min=0;
                    hour++;
                }
            }
}
```

第 5 章

1. 以下叙述中不正确的是 _____。

A、在方法中 , 通过 return 语句传回方法值

B、在一个方法中，可以执行有多条 return 语句，并返回多个值

C、在 Java 中 , 主方法 main（）后的一对圆括号中也可以带有参数

D、在 Java 中 , 调用方法可以在 System.out.println() 语句中完成

B

2. 以下正确的描述是 _____。

A、方法的定义不可以嵌套，但方法的调用可以嵌套

B、方法的定义可以嵌套，但方法的调用不可以嵌套

C、方法的定义和方法的调用均不可以嵌套

D、方法的定义和方法的调用均可以嵌套

A

3. 以下正确的说法为 _____。

A、在不同方法中不可以使用相同名字的变量

B、实际参数可以在被调方法中直接使用

C、在方法内定义的任何变量只在本方法范围内有效

D、在方法内的复合语句中定义的变量只在本方法语句范围内有效

C

4. 按 Java 语言的规定，以下正确的说法是 _____。

A、实参不可以是常量，变量或表达式

B、形参不可以是常量，变量或表达式

C、实参与其对应的形参占用同一个存储单元

D、形参是虚拟的，不占用存储单元

D

5. 一个 Java Application 程序中有且只有一个 _____方法，它是整个程序的执行入口。

main（）方法

6. 方法通常可以认为由两部分组成，它们是 _____和 _____。

方法头和方法体

7. 读程序写结果。

```
public class   Test {
  static void m(int x, int y, int z)
  { x=111;    y=222;    z=333;
  }
  public static void main(String args[ ] )
  { int x=100, y=200, z=300;
     m(x, y, z);
     System.out.println( “x= ”+x+ ”y= ”+y+ ”z= ”+z);
  }
}
```

x=100y=200z=300

8. 编写一个判断某个整数是否为素数的方法。

```
public boolean prime(int x)
        {
                for(int j=2;j<x;j++)                    //检验是否满足素数条件
                    if (x%j==0)                 //不满足
                        return false;
                return true;
                }
```

9. 编写两个方法，分别求两个整数的最大公约数和最小公倍数，在主方法中由键盘输入两个整数并调用这两个方法，最后输出相应的结果。

```
import java.io.*;
public class Test
{       public static void main(String[] args)
            {int a=0;
                System.out.print(" 请输入数   a:");
                try{
                    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                    a=Integer.parseInt(br.readLine());
               }catch(IOException e){}
               int b=0;
                System.out.print(" 请输入数   b:");
               try
           {BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                    b=Integer.parseInt(br.readLine());
                    }catch(IOException e){}
               if(a<=0||b<=0)
               System.out.println(" 输入不合法  !");
               System.out.println(" 最大公约数为   :"+Gys(a,b));
               System.out.println(" 最小公倍数为   :"+Gbs(a,b));
            }
        public static int Gys(int a,int b)
        {
                int r=0;
                if(a<b)
                {r=a;
                  b=a;
```

```
  a=r;}
int i=1;
while(i!=0)
{i=a%b;
  a=b;
  b=i;
  }
        return a;
}
public static int Gbs(int a,int b)
{
          int ab=a*b;
        int r=0;
        if(a<b)
        {r=a;
          b=a;
          a=r;}
        int i=1;
        while(i!=0)
                {i=a%b;
                  a=b;
                  b=i;
                  }
                return ab/a;
        }
}
```

10. 以下程序执行后的输出为 _____。

```
    public class Test
    {
        static int m1(int a ,int b)
    {
        int c;
        a+=a;
        b+=b;
        c=m2(a,b);
        return(c*c);
    }
```

```
 static int m2( int a,int b)
 {
        int c;
        c=a*b%3;
        return( c );
 }
public static void main(String[] args)
      {
          int x=1,y=3,z;
          z= m1(x,y);
          System.out.println("z="+z);
       }
    }

        z=0
```

11. 编写一个方法，实现求某整数的各位数字之和的功能。

```
import java.io.*;
public class Test
{      public static void main(String[] args)
        { int i,a=0,s=0;
          System.out.print(" 请输入数   a:");
          try{
              BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
              a=Integer.parseInt(br.readLine());
          }catch(IOException e){}
          for( i=10;a%i!=a;i*=10)
          {
              s+=(a%i)/(i/10);
          }
          s+=a/(i/10);
          System.out.println(" 各位数之和为   "+s);
        }
}
```

12. 编写完成十进制整数到八进制的转换方法。

```
import java.io.*;
public class Test
```

```
{       public static void main(String[] args)
            { int i,k,a=0,d,s=0;
                System.out.print(" 请输入数 :");
                try{
                    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                    a=Integer.parseInt(br.readLine());
                }catch(IOException e){}
                d=a%8;
                s+=d;
                k=10;
                for(i=64;a-d!=0;i*=8)
                {
                    s+=(a-d)%i*8/i*k;
                    k*=10;
                    d=a%i;
                }
                System.out.println(" 八进制数为 "+s);
            }
}
```

13. 用于指出数组中某个元素的数字被叫做 _____；数组元素之所以相关，是因为它们具有相同的 _____和 _____。

下标；数组名和数据类型。

14. 数组 **int results[ ] = new int[6]** 所占存储空间是 _____字节。

24

15. 使用两个下标的数组被称为 _____数组，假定有如下语句：

float scores[ ][ ] = { {1， 2， 3}，{4， 5}，{6， 7， 8， 9} }；

则 **scores.length**的值为： _____， **scores[1].length**的值为： _____，

scores[1][1]的值为： _____。

二维； 3； 2； 5。

16. 从键盘上输入 **10** 个双精度浮点数后，求出这 **10** 个数的和以及它们的平均值。要求分别编写相应求和及求平均值的方法。

```
import java.io.*;
```

```
public class Test
{
    public static void main(String[] args)throws IOException
    {
        InputStreamReader reader=new InputStreamReader(System.in);
        BufferedReader input=new BufferedReader(reader);
        String temp;
      double x[]=new double[10];
      for(int i=0;i<10;i++)
        {
            temp=input.readLine();
            x[i] = Double.parseDouble(temp);
         }
        S(x);
        Avg(x);
}
public static void S(double x[])
{
    //计算并输出和
      double sum=0;
      for(int i=0;i<10;i++)
      {
                sum+=x[i];
      }
          System.out.println(" 总和： "+sum);
  }

public static void Avg(double x[])
{
     //计算并输出平均数
    double sum=0;
              for(int i=0;i<10;i++)
               {
                        sum+=x[i];
                }
                System.out.println(" 平均数： "+sum/10);
              }
}
```

17. 利用数组输入 **6** 位大学生 **3** 门课程的成绩，然后计算

（**1**）每个大学生的总分；

（**2**）每门课程的平均分；

```
import java.io.*;
public class Scores
{
public static void main(String[] args)throws IOException
{
    int sum=0;      //总分
    InputStreamReader reader=new InputStreamReader(System.in);
    BufferedReader input=new BufferedReader(reader);
       int x[][]=new int[6][3];
       //录入成绩
    for(int i=0;i<6;i++)
       {
         for (int j=0;j<3 ;j++ )
           {
                    System.out.print((i+1)+"  号同学 "+(j+1)+"  号课程分数  ");
              temp=input.readLine();
              x[i][j] = Integer.parseInt(temp);
           }
       }
//计算并输出每一位同学的总分
for(int i=0;i<n;i++)
 {
    for (int j=0;j<5 ;j++ )
    {
            sum+=x[i][j];
    }
       System.out.println((i+1)+"  号同学总分：  "+sum);
          sum=0;
   }
     //计算并输出每一门课程的平均分
for(int i=0;i<5;i++)
   {
      for (int j=0;j<n ;j++ )
      {
```

```
                    sum+=x[j][i];
                }
                System.out.println((i+1)+"  号课程班级平均分：    "+sum*1.0/n);
                sum=0;
            }
        }
}
```

18. 编写一个方法，实现将字符数组倒序排列，即进行反序存放。

```
import java.io.*;
public class Test
{
    public static void main(String[] args)throws IOException
    {
        char c[] = {'O','l','y','m','p','i','c',' ','G','a','m','e','s'};
        rever(c);
        System.out.println(c);
        }
        public static void rever(char c[])
        {
            char t;
        for(int i=0,j=c.length-1;i<j;i++,j--)
        {
            t=c[i];
            c[i]=c[j];
            c[j]=t;
        }
    }
}
```

19. **Java** 语言为什么要引入方法这种编程结构？

提高复用度，减少程序代码量

促进程序结构化，提高可读性和可维护性

20. 为什么要引入数组结构，数组有哪些特点，**Java** 语言创建数组的方式有哪些？

为了便于处理一批同类型的数据，Java 语言引入了数组类型；

首先，数组中的每个元素都是相同数据类型的；其次，数组中的这些相同数据类型元素是通过数组下标来标识的，并且该下标是从 0 开始的；最后，数组元素在内存中的存放是连

续的。

Java 语言规定，创建数组可以有两种方式：初始化方式和 new 操作符方式。初始化方式是指直接给数组的每一个元素指定一个初始值，系统自动根据所给出的数据个数为数组分配相应的存储空间，通常这样创建数组的方式适用于数组元素较少的情形。对于数组比较大的情形，即数组元素过多，用初始化方式显然不妥，这时应采用第二种方式，即 new 操作符方式。

第 6 章

1. 实现类 **MyClass** 的源码如下：

```
class MyClass extends Object{
        private int x;
        private int y;
        public MyClass( ){
            x=0;
            y=0;
        }
        public MyClass(int x, int y){
            ... ... ...
        }
        public void show( ){
            System.out.println("\nx="+x+"       y="+y);
        }
        public void show(boolean flag){
            if (flag) System.out.println("\nx="+x+"       y="+y);
            else System.out.println("\ny="+y+"      x="+x);
        }
        protected void finalize( ) throws throwable{
            super.finalize();
        }
}
```

在以上的源代码中，类 **MyClass** 的成员变量是 _____；构造方法是 _____；对该类的一个实例对象进行释放时将调用的方法是 _____。（多选）

（**A**） **private int x;**

（**B**） **private int y;**

（**C**） **public MyClass( )**

（**D**） **public MyClass(int x, int y)**

（**E**） **public void show( )**

（**F**） **public void show (boolean flag)**

（**G**） **protected void finalize( ) throws throwable**

答案： AB CD G

2. 上题所声明的类 **MyClass** 的构造方法 **MyClass(int x, int y)** 的目的是使 **MyClass** 的成员变量 **private int x, private int y** 的值分别等于方法参数表中所给的值 **int x, int y** 。请写出 **MyClass(int x, int y)** 的方法体（用两条语句） ：

_____;

答案： this.x=x this.y=y

3. **MyClass** 声明同第一题。

设 **public static void main(String args[])** 方法体如下：

```
{
    MyClass myclass;
    myclass.show();
}
```

编译运行该程序将会有何结果？

（**A**） **x=0 y=0**

（**B**） **y=0 x=0**

（**C**） **x=... y=...** （**x**，**y** 具体为何值是随机的）

（**D**）源程序有错

答案： D

4. **MyClass** 声明同第一题。

设 **public static void main(String args[])** 方法体如下：

```
{
    MyClass myclass=new MyClass(5,10);
    myclass.show(false);
}
```

编译运行该程序将会有何结果？

（**A**） **x=0 y=0**

（**B**） **x=5 y=10**

（**C**） **y=10 x=5**

（**D**） **y=0 x=0**

答案： C

5. **MyClass** 声明同第一题。

设 **public static void main(String args[])** 方法体如下：

```
{
    MyClass myclass=new MyClass(5,10);
    myclass.show(false);
}
```

现在想在 **main** 方法中加上一条语句来释放 **myclass**对象，应用下面哪条？

（**A**） **myclass=null;**

（**B**） **free(myclass);**

（**C**） **delete(myclass);**

（**D**） **Java** 语言中不存在相应语句

答案： A

6. 假设已编写好了类 **Class1:**

package mypackage; public class Class1{ … … }

它存在 **Class1.java** 文件中。

现在 **main** 方法所在的源程序 **MainPro.java** 如下：

```
import mypackage;
    &hellip; &hellip;
```

假设操作系统中的 **CLASSPATH** 环境变量已被设成 **"c:\java\lib\classes.zip;.; "**，而 **main** 方法所在的源程序 **MainPro.java** 存在目录 **c:\mydir** 中（当前工作目录为 **c:\mydir**），那么 **Class1.class**文件应存放在那个目录中呢？ _____

答案： c: \mydir\ mypackage

7. **MyClass** 声明同第一题。

设程序如下：

```
class MyClass extends Object{.....}
public class MyPro{
    public static void main(String args[]){
        MyClass myclass=new MyClass(5,10);
        System.out.println("\nx="+myclass.x+"      y="+myclass.y);
    }
}
```

编译运行结果是什么？

（**A**） **x=0 y=0**

（**B**） **x=5 y=10**

（**C**）编译不能通过

答案： C

8. 接口中可以有的语句为 _____；（从 **ABCD** 中多选）

一个类可以继承 _____父类，实现 _____接口；一个接口可继承 _____接口；（从 **EF** 中单选）

接口 _____继承父类， _____实现其它接口；实现某个接口的类 _____当作该接口类型使用；（从 **GH** 中单选）

（**A**） **int x;**

（**B**） **int y=0;**

（**C**） **public void aa( );**

（**D**） **public void bb( ){System.out.println("hello");}**

（**E**）仅一个

（**F**）一个或多个

（**G**）可以

（**H**）不可以

答案： BC ； E ； F ； F ； H ； H ； G

9. 定义一个表示学生的类 **student**，成员变量有学号、姓名、性别、年龄，方法有获得学号、姓名、性别、年龄；修改年龄。书写 **Java** 程序创建 **student** 类的对象及测试其方法的功能。

答：

```
public class student{
    private int stu_ID;
    private String name;
    private String sex;
    private int old;
    student(int id,String name,String sex,int old){
        stu_ID=id;
        this.name=name;
        this.sex=sex;
        this.old=old;
    }
    void show_id(){
        System.out.println("the student ID is:"+stu_ID);
    }
    void show_name(){
        System.out.println("the student name is:"+name);
    }
    void show_sex(){
        System.out.println("the student sex is:"+sex);
    }
    void show_old(){
        System.out.println("the student old is:"+old);
    }
    void change_old(int newyear){
        old=newyear;
```

```
    }
    public static void main(String args[]){
        student Lee=new student(,"Li Ming","M",18);
        Lee.show_id();
        Lee.show_name();
        Lee.show_sex();
        Lee.show_old();
        Lee.change_old(20);
        Lee.show_old();
    }
}
```

10. 根据下面的要求编程实现复数类 **Complex**。

(1)复数类 **Complex** 的属性：

real 代表复数的实数部分

imagin 代表复数的虚数部分

(2) 复数类 **Complex** 的方法：

Complex( )：构造函数，将实部、虚部都置为 **0**；

Complex(double r,double i):构造函数， 创建复数对象的同时完成复数的实部、虚部的初始化， **r** 为实部的初值， **i** 为虚部的初值；

getReal( )：获得复数对象的实部；

getImagin( )：获得复数对象的虚部；

complexAdd(Complex Number) ：当前复数对象与形参复数对象相加，所得的结果也是复数值，返回给此方法的调用者；

complexMinus(Complex Number) ：当前复数对象与形参复数对象相减，所得的结果也是复数值，返回给此方法的调用者；

complexMulti(Complex Number)：当前复数对象与形参复数对象相乘，所得的结果也是复数值，返回给此方法的调用者；

toString( )：把当前复数对象的实部、虚部组合成 **a+bi** 的字符串形式，其中 **a** 和 **b** 分别为实部和虚部的数据。

答案：

```
public class Complex{
    private double real;
    private double imagin;
    Complex(){
        real=0;
        imagin=0;
    }
    Complex(double r,double i){
        real=r;
        imagin=i;
```

```
}
public double getReal(){
        return real;
}
public double getImagin(){
        return imagin;
}
public Complex complexAdd(Complex Number){
        real=real+Number.real;
        imagin=imagin+Number.imagin;
        return this;
}
public Complex complexMinus(Complex Number){
        real=real-Number.real;
        imagin=imagin-Number.imagin;
        return this;
}
public Complex complexMulti(Complex Number){
        real=real*Number.real;
        imagin=imagin*Number.imagin;
        return this;
}
        public void tostring(){
                System.out.println(real+"+"+imagin+"i");
        }
        public static void main(String args[]){
                Complex a=new Complex();
                Complex b=new Complex(3,4);
                Complex c=new Complex(2,3);
                a.tostring();
                b.tostring();
                c.complexAdd(b);
                c.tostring();
        }
}
```

11. 解释 **this** 和 **super** 的意义和作用。

答： Java 中， this 用来引用当前对象，与 this 类似， super 用来引用当前对象的父类。

12. 什么是继承？继承的意义？如何定义继承关系？

答：继承是一种由已有的类创建新类的机制。

通过继承可以实现代码的复用， 使程序的复杂性线性地增长， 而不是随规模增大呈几何级数增长。

由于父类代表了所有子类的共性，　而子类既可继承其父类的共性，　又可以具有本身独特的个性，在定义子类时，只要定义它本身所特有的属性与方法就可以了。

13．什么是多态？　**Java** 程序如何实现多态？有哪些实现方式？

答：多态性是指同名的不同方法在程序中共存。　即为同一个方法定义几个版本，　运行时根据不同情况执行不同的版本。　调用者只需使用同一个方法名，　系统会根据不同情况，　调用相应的不同方法，从而实现不同的功能。多态性又被称为“ 一个名字，多个方法 ”　。

多态性的实现有两种方式：覆盖实现多态性、重载实现多态性。

14．利用多态性编程，实现求三角形、正方形和圆形面积。方法：抽象出一个共享父类，定义一函数为求面积的公共界面，再重新定义各形状的求面积函数。在主类中创建不同类的对象，并求得不同形状的面积。

答：利用多态性编程，　实现求三角形、　正方形和圆形面积。　方法：抽象出一个共享父类，定义一函数为求面积的公共界面，　再重新定义各形状的求面积函数。　在主类中创建不同类的对象，并求得不同形状的面积。

```
abstract class Shape
{
    abstract float area();
}class Circle extends Shape
{
    public float r;
    Circle(float r)
    {
        this.r = r;
     }
     public float area()
     {
        return 3.14*r*r;
      }
}class Rectangle    extends Shape
{
    public float width,height;
    Rectangle (float w, float h)
    {
        width = w;
        height = h;
    }
    public float area()
    {
        return width*height;
    }
}
```

第 7 章

1. String 类型与 **StringBuffer** 类型的区别是什么？

答： String 类型的字符串是对原字符串的拷贝进行操作，而 StringBuffer 类型的字符串是对 原字符串 本身进行操作的，操作后的结果会使原字符串发生改变。

2. 有如下四个字符串 **s1、s2、s3**和 **s4:**

String s1="Hello World! ";

String s2=new String("Hello World! ");

s3=s1;

s4=s2;

求下列表达式的结果是什么？

s1==s3

s3==s4

s1==s2

s1.equals(s2)

s1.compareTo(s2)

答：下列表达式的结果是：

false

false

false

true

0

3. 下面程序输出的结果是什么？

```
public class Test {
    public static void main(String[] args) {
        String s1="I like cat";
        StringBuffer sb1=new StringBuffer ("It is Java");
        String s2;
        StringBuffer sb2;
        s2=s1.replaceAll("cat","dog");
        sb2=sb1.delete(2,4);
        System.out.println("s1 为： "+s1);
        System.out.println("s2 为： "+s2);
        System.out.println("sb1 为： "+s1);
```

```
        System.out.println("sb2 为： "+s2);
    }
}
```

答：程序的输出结果为：

s1为： I like cat

s2为： I like dog

sb1为： I like cat

sb2为： I like dog

4. 设 **s1**和 **s2**为 **String** 类型的字符串， **s3**和 **s4**为 **StringBuffer** 类型的字符串， 下列那个语句或表达式不正确？

s1="Hello World! ";

s3="Hello World! ";

String s5=s1+s2;

StringBuffer s6=s3+s4;

String s5= s1-s2;

s1<=s2

char c=s1.charAt(s2.length());

s4.setCharAt(s4.length(),'y');

答：语句或表达式不正确的有：

s3="Hello World! ";

StringBuffer s6=s3+s4;

String s5= s1-s2;

s1<=s2

5. **StringTokenizer** 类的主要用途是什么？该类有哪几种重要的方法？它们的功能是什么？

答： StringTokenizer 类的主要用途是可以通过分析一个字符串把字符串分解成可被独立使用的单词。常用的方法有如下几种：

public String nextToken();

功能：逐个获取字符串中的单词并返回该字符串。

public String nextToken(String delim)

功能：以 delim 作为分隔符逐个获取字符串中的单词并返回该字符串。

public int countTokens()

功能：返回单词计数器的。

public boolean hasMoreTokens();

功能：检测字符串中是否还有单词，如果还有单词，则返回 true，否则返回 false。

6. 下列程序输出的结果是什么？

import java.util.*;

```
public class Hello {
        public static void main(String[] args) {
                String s="Friday;Saturday\Sunday Monday,Tuesday";
                StringTokenizer stk=new StringTokenizer(s,"; \");
                while(stk.hasMoreTokens()){
                        System.out.println(stk.nextToken());
                }
        }
}
```

解：输出的结果是：

Friday

Saturday

Sunday

Monday,Tuesday

7. 编写程序，在命令行输入 **java** 类文件名 **11 24 62 73 103 56** ，求这一串数字的最大值和平均数。

解：

```
        public class Hello {
                public static void main(String[] args) {
                        double total=0;
                        int max=0;
                        for(int i=0;i<args.length;i++){
                                total=total+Integer.parseInt(args[i]);
                                if(max < Integer.parseInt(args[i]))
                                        max=Integer.parseInt(args[i]);
                        }
                        System.out.println(max);
                        System.out.println(total/args.length);
                }
        }
```

8. 编写程序，输入两个字符串，完成以下几个功能：

（1）求出两个字符串的长度。

（2）检验第一个串是否为第二个串的子串。

（3）把第一个串转化为 byte 类型并输出。

解：

```
        import java.lang.System;
        import java.util.*;
        public class Hello {
                public static void main(String[] args) {
```

```
Scanner scan = new Scanner(System.in);
System.out.println(" 请输入字符  1");
String str1 = scan.nextLine();
System.out.println(" 请输入字符  2");
String str2 = scan.nextLine();
if(str2.indexOf(str1)==-1)
        System.out.println(" 字符串 1不是字符串  2的子串 ");
else
        System.out.println(" 字符串 1是字符串 2的子串 ");
                byte b[ ]= str1.getBytes();
                for(int i=0;i<str1.length();i++)
                System.out.println(b[i]);
        }
}
```

第 8 章

1. Java 为什么要引入线程机制？线程的概念是什么？线程和进程的区别是什么？解释什么是 Java 的多线程？

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。

所谓线程是指进程中单一顺序的执行流。

进程是一个动态执行的程序 ,当你运行一个程序的时候，就创建了一个用来容纳组成代码和数据空间 的进程。每一个进程都有自己的一块内存空间和一组系统资源，它们之间都是独立的。 线程可以共享内存单元和 系统资源 ，但不能够单独执行， 必须存在于某个进程当中。它是比进程更小的能独立运行的基本单位。

Java 中的线程由虚拟的 CPU、CPU 所执行的代码和 CPU 所处理的数据三部分组成。 Java 的多线程就是系统每次给 Java 程序一个 CPU 时间， Java 虚拟处理机在多个线程之间轮流切换，保证每个线程都能机会使用 CPU 资源，不过每个时刻只能有一个线程在执行。

2. 线程创建方式有哪两种？请举例说明。

答：两种途径来实现多线程的创建：一种是直接继承 Thread 类并重写其中的 run( ) 方法 ， 另一种是使用 Runnable 接口。

途径一：

```
class SimpleThread extends Thread {
        private String threadname;
        public SimpleThread(String str) {
            threadname=str;
```

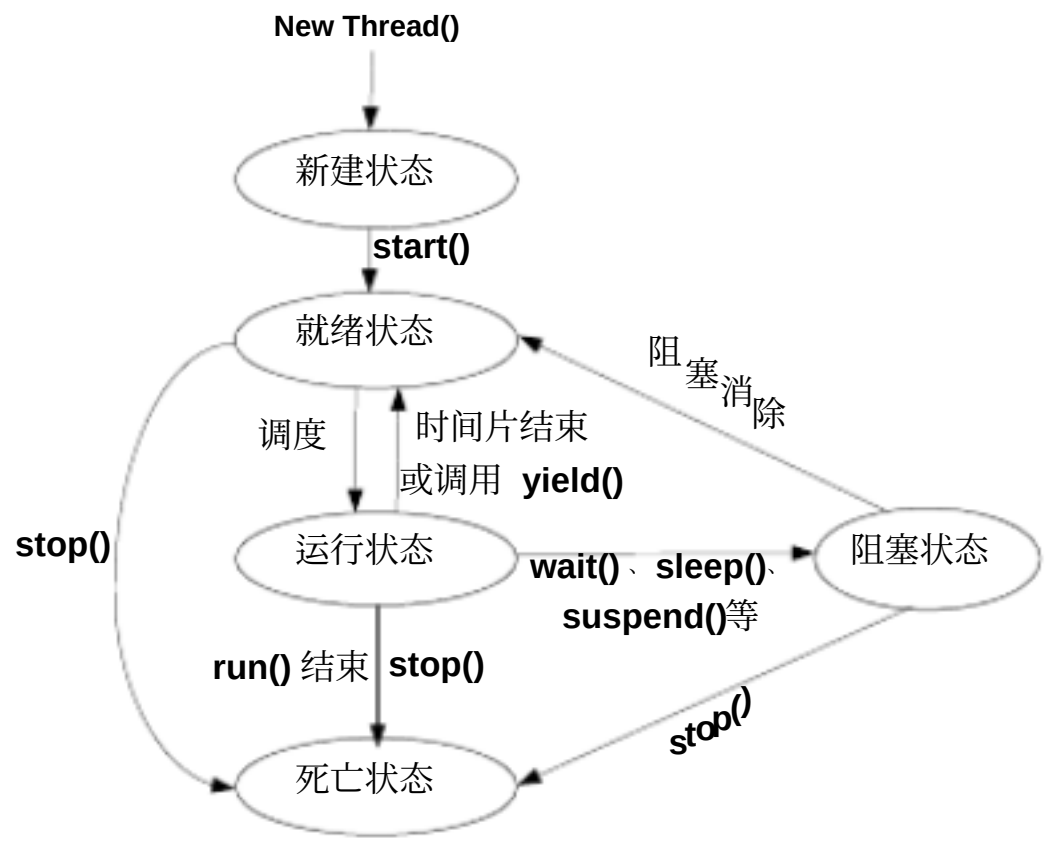
```
}
public void run() {
    for (int i = 0; i < 10; i++) {
        System.out.println(threadname+" 被调用！ ");
        try {
                sleep(10);
        } catch (InterruptedException e) {}
        }
    }
}
```

途径二：

```
class SimpleThread implements Runnable {
        public SimpleThread(String str) {
        super（str）;
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(getName()+" 被调用！ ");
            try {
                Thread.sleep(10);
            } catch (InterruptedException e) {}
        }
    }
}
```

3. 什么是线程的生命期？它包括哪几种状态？它们的关系是什么？

答：线程的生命期是指从线程被创建开始到死亡的过程，通常包括 5 种状态：新建、就绪、运行、阻塞、死亡。它们的关系如图所示：



4. 请举例说明如何实现线程的同步（用两种方法）　　。

方法一（方法同步）　：

```
class Stack{
        private int number;
        private int     len=0;
        public synchronized void put(int t){
                while(len == 1){
                        try{
                                this.wait();
                        }catch(InterruptedException e){}
                }
                number=t;
                len=1;
                this.notify();
        }
        public synchronized int get(){
                while(len ==0){
                        try{
                                        this.wait();
                        }catch(InterruptedException e){}
                }
                        len=0;
                        this.notify();
                        return number;
                }
}
```

方法二（对象同步）：

```
class Stack{
        private int number;
        private int     len=0;
```

```
public void put(int t){
        synchronized (this){
                while(len == 1){
                        try{
                                this.wait();
                        }catch(InterruptedException e){}
                }
                number=t;
                len=1;
                this.notify();
        }
}
public int get(){
        synchronized (this){
                while(len ==0){
                        try{
                                        this.wait();
                        }catch(InterruptedException e){}
                }
                len=0;
                this.notify();
                return number ;
                }
         }
}
class Input implements Runnable{
        Stack sstack;
        public Input(Stack s){
                sstack=s;
        }
        public void run(){
                for(int i=1;i<5;i++){
                        sstack.put(i);
                        System.out.println(" 向 Stack放入数字 :"+i);
                        try{
                                        Thread.sleep((int)(Math. random ()*10));
                        }catch(InterruptedException e){}
                }
        }
}
class Output implements Runnable{
        Stack sstack;
        public Output(Stack s){
                sstack=s;
```

```
}
public void run(){
        int temp;
        for(int i=1;i<5;i++){
                temp=sstack.get();
                System.out.println(" 向 Stack取出数字 :"+temp);
                try{
                        Thread.sleep((int)(Math. random ()*10));
                }catch(InterruptedException e){}
                }
        }
}
public class Hello {
        public static void main(String args[]){
        Stack s = new Stack();
        Input t=new Input(s);
        Output o = new Output(s);
        Thread t1 = new Thread (t);
        Thread t2 = new Thread (o);
        t1.start();
        t2.start();
        }
}
```

5. **Java** 种有哪些情况会导致线程的不可运行？

答：一是调用了 wait（）方法，使得线程等待；二是调用了 sleep（int time）方法，使得线程休眠；三是调用了 suspend（）方法，使得线程挂起；四是由于输入输出流而引起阻塞。

6. **wait()** 方法和 **sleep()**方法的区别是什么？

答： sleep()方法使线程进入睡眠状态，但它并不会释放线程持有的资源，不能被其他资源唤醒，不过睡眠一段时间会自动醒过来，而 wait() 方法让线程进入等待状态的同时也释放了持有的资源，能被其他资源唤醒。

7. 线程组的作用是什么？如何创建一个线程组？

答：线程组是把多个线程集成到一个对象里并可以同时管理这些线程。

在线程创建时， 可以将线程放在某个指定的线程组中， 也可以将它放在一个默认的线程组。若创建线程而不明确指定属于哪个组， 它们就会自动归属于系统默认的线程组。 以下三种 Thread 类的构造方法实现线程创建的同时指定其属于哪个线程组。

public Thread （ThreadGroup group， Runnable target）；

public Thread （ThreadGroup group， String name）；

public Thread （ThreadGroup group， Runnable target， String name）；

8. **Java** 线程调度的原则是什么？

答：Java 调度器调度遵循以下原则：优先级高的线程比优先级低的线程线程先调度，优先级相等的线程按照排队队列的顺序进行调度，先到队列的线程先被调度。当一个优先级低的线程运行过程中，来了一个高优先级线程，在时间片方式下，优先级高的线程要等优先级低的线程时间片运行完毕才能被调度，而在抢占式调度方式下，优先级高的线程可以立刻获得 CPU 的控制权。

9. 如何理解死锁？

答：如果两个或多个线程都在互相等待对方持有的锁（唤醒），那么这些线程都进入阻塞状态，永远地等待下去，无法执行，程序就出现了死锁。

10. 下列程序输出的结果是什么？

```
class Daemon extends Thread {
        public void run() {
                if(this.isDaemon()==false)
                        System.out.println("thread is not daemon");
                else
                        System.out.println("thread is daemon");
                try {
                        for(int i=0;i<10;i++){
                                System.out.println(i);
                                Thread.sleep(200);
                        }
                    }catch (InterruptedException e){}
                System.out.println("thread done!");
        }
 }
public class Test {
        public static void main (String[] args) {
                Thread t=new Daemon();
                t.setDaemon(true);
                t.start();
                try {
                        Thread.sleep(900);
                    }catch (InterruptedException e){}
                System.out.println("main done!");
        }
}
```

答：输出结果为：　thread is daemon

0

1

2

3

4

main done!

11. 编写程序实现如下功能：　一个线程进行如下运算　**1*2+2*3+3*4+ …… +1999*2000,** 而另一个线程则每隔一段时间读取前一个线程的运算结果。

解：

```
class Result{
        private long total=1*2;
        public     void write(long t){
                total=total+t;
        }
        public     long read(){
                return total;
         }
}
class Addition implements Runnable{
        Result rt;
public Addition(Result s){
        rt=s;
}
public void run(){
        for(int i=2;i<2000;i++){
                long tmp=0;
                tmp=i*(i+1);
                rt.write(tmp);
                try{
                                Thread.sleep(10);
                }catch(InterruptedException e){}
        }
}
}
class Output implements Runnable{
        Result rt;
        public Output(Result s){
                rt=s;
        }
        public void run(){
                long temp;
                for(int i=1;i<200;i++){
```

```
            temp= rt.read();
                System.out.println(temp);
                try{
                    Thread.sleep(100);
                }catch(InterruptedException e){}
            }
        }
    }
    public class Hello {
        public static void main(String args[]){
        Result rt = new Result();
        Addition t=new Addition(rt);
        Output o = new Output(rt);
        Thread t1 = new Thread (t);
        Thread t2 = new Thread (o);
        t1.start();
        t2.start();
        }
    }
```

12. 编写程序实现如下功能：第一个线程打印 **6** 个 **a**，第二个线程打印 **8** 个 **b**，第三个线程打印数字 **1** 到 **10**，第二和第三个线程要在第一个线程打印完成之后才能开始打印。

解：

```
    class Thread1 extends Thread
    {
        public void run() {
            for(int i=0 ; i<6 ; i++) {
                System.out.println('a');
            }
        }
    }
    class Thread2 extends Thread
    {
        public void run() {
            try{
                for(int i=0 ; i<8 ; i++) {
                    System.out.println('b');
                    Thread.sleep(100);
                }
            }catch(InterruptedException e){}
        }
    }
    class Thread3 extends Thread
    {
```

```
        public void run() {
            try{
                for(int i=1 ; i<=10 ; i++) {
                    System.out.println(i);
                    Thread.sleep(100);
                }
            }catch(InterruptedException e){}
        }
    }

    public class Test
    {
        public static void main (String[] args) {
            Thread1 t1 = new Thread1();
            Thread2 t2 = new Thread2();
            Thread3 t3 = new Thread3();
            t1.start() ;
            try{
                t1.join() ;
            }catch(InterruptedException e) {}
            t2.start() ;
            t3.start();
        }
    }
```

13. 编写一个线程同步的程序：有一个字符缓冲区，长度为 **length**，我们创建两个线程，其中一个线程向字符缓冲区写入一个字符， （字符缓冲区一次只能装入一个字符） ，另一个线程从字符缓冲区取出一个字符，并且输出，要保证当一个线程在写字符的时候，另一个线程不能访问字符缓冲区，而且在字符缓冲区为空的时候取不出字符，而在字符缓冲区满的时候写不进字符。

解：

```
    class Stack{
        private char[] stack =new char[length+1];
        private int num=0;
        public synchronized void write(char c){
        while(num == length){
            try{
                this.wait();
            }catch(InterruptedException e){}
        }
        num++;
        stack[num]=c;
        this.notify();
```

```
}
public synchronized char read(){
        while(num ==0){
                try{
                        this.wait();
                }catch(InterruptedException e){}
        }
        num--;
        this.notify();
                return stack[num+1];
         }
}
class Input implements Runnable{
        Stack sstack;
        public Input(Stack s){
                sstack=s;
        }
        public void run(){
                for(int i=1;i<10;i++){
                        char tmp;
                        tmp=(char)(96+i);
                        sstack.write(tmp);
                        System.out.println(" 向字符缓冲区写入字符    :"+tmp);
                        try{
                                Thread.sleep((int)(Math. random ()*10));
                        }catch(InterruptedException e){}
                }
        }
}
class Output implements Runnable{
        Stack sstack;
        public Output(Stack s){
                sstack=s;
        }
        public void run(){
                char temp;
                for(int i=1;i<10;i++){
                        temp=sstack.read();
                        System.out.println(" 从字符缓冲区取出字符    :"+temp);
                        try{
                                Thread.sleep((int)(Math. random()*10));
                        }catch(InterruptedException e){}
                }
        }
}
```

```
}
public class Test {
        public static void main(String args[]){
        Stack s = new Stack();
        Input t=new Input(s);
        Output o = new Output(s);
        Thread t1 = new Thread (t);
        Thread t2 = new Thread (o);
        t1.start();
        t2.start();
        }
}
```

第 9 章

1. 什么是 **Java Applet** 程序，它与前面介绍过的 **Java Application** 有何不同？

Applet 一般称之为小应用程序， Java Applet 就是用 Java 语言编写的这样的一些小应用程序，它们可以通过嵌入到 Web 页面或者其他特定的容器中来运行，也可以通过 java 开发工具的 appletviewer 来运行，并能够产生特殊的效果。

与独立执行的 Java 应用程序不同， Applet 有自己的一套执行流程，而不是通过 main 方法来开始执行程序，并且在运行过程中 Applet 常会与用户进行互动操作，显示动态的页面效果，并且还会进行严格的安全检查，以防止潜在的不安全因素。

2. 简述 **Java Applet** 程序的开发步骤。

Applet 的开发步骤大致可以分为以下三个步骤：

（1）用 UltraEdit 或 Notepad 等纯文本软件编辑 Java Applet 源程序。

（2）利用 javac 编译器将 Applet 源程序转换成 class 字节码文件。

（3）编写 HTML 页面，并通过＜ APPLET ＞＜ /APPLET ＞标签引用上述字节码文件。

3. 简述与 **Java Applet** 生命周期相关的四个方法。

Applet 小程序的生命周期相对于 Application 而言较为复杂。在其生命周期中涉及到 Applet 类的四个方法： init （）、 start（）、 stop（）和 destroy（）， Applet 的生命周期中有相对应的四个状态： 初始态、 运行态、 停止态和消亡态。 当程序执行完 init（）方法以后， Applet 小程序就进入了初始态；然后立刻执行 start（）方法， Applet 小程序进入运行态；当 Applet 小程序所在的浏览器图标化或者是转入其它页面时，该 Applet 小程序立刻执行 stop （）方法，使 Applet 小程序进入停止态；在停止态中，如果浏览器又重新加载该 Applet 小程序所在的页面，或者是浏览器从图标中还原，则 Applet 小程序又会调用 start（）方法，进入运行态；不过，在停止态时，若浏览器被关闭，则 Applet 小程序会调用 destroy（）方法，使

其进入消亡态。

4. 编写一个 **Java Applet** 程序，使其在窗口中以红色、绿色和蓝色为顺序循环显示字符串：“**Welcome to Java Applet**”。

```
import java.awt.*;
import java.awt.Color;
import java.applet.Applet;
public class JumpText extends Applet implements Runnable{
    Thread runThread;
    int i;
    public void start(){
        if(runThread==null){
            runThread=new Thread(this);
            runThread.start();
            i=0;
        }
    }
    public void stop(){
        if(runThread!=null){
            runThread.stop();
            runThread=null;
        }
    }
    public void run(){
        while(true){
            i=(i+1)%3;
            repaint();
            try{
                Thread.sleep(1000);
            }catch(InterruptedException e){}
        }
    }
    public void paint(Graphics g){
        switch (i){
            case 0: g.setColor(Color.red); break;
            case 1: g.setColor(Color.green); break;
            case 2: g.setColor(Color.blue); break;
        }
```

```
        g.drawString("Welcome    to    Java    Applet",8,50);
    }
}
```

5. 列举几个 **Graphics** 类提供的方法，并说明其用法。

（1）drawString （String str， int x， int y） 用于字符串输出

（2）public void drawLine （int x1， int y1， int x2， int y2）

其功能为以像素为单位绘制一条从（ x1， y1）至（ x2， y2）的直线。

（3）drawRect （）方法可以进行矩形的绘制

（4）fillRect （）方法用于绘制以前景色填充的实心矩形

（5）drawRoundRect （）和 fillRoundRect （）方法来绘制圆角矩形

（6）getImage（）和 drawImage （）方法来实现在程序中显示漂亮的背景或图像

（7）getAudioClip （）、 loop （）和 stop（）等方法用来支持声音文件的播放

6. 编写 **Applet** 程序，绘制一幅五颜六色的图。

```
import java.awt.*;
import java.applet.*;
public class ColorApplet extends Applet
{
  public void paint(Graphics g)
{
        g.setColor(Color.red);
        g.drawOval(10,10,30,30);
        g.setColor(Color.green);
        g.drawOval(50,50,80,80);
        g.setColor(Color.blue);
        g.drawOval(110,110,130,130);
        g.setColor(Color.yellow);
        g.drawOval(110,10,30,30);
        g.setColor(Color.pink);
        g.drawOval(160,50,80,80);
    }
}
```

7. 编写一简易自行车在公路上由左向右行驶的 **Applet** 程序。

```
import java.awt.*;
import java.awt.Color;
import java.applet.Applet;
public class MovingImg extends Applet{
```

```
Image img0,img1;
int x=10;
public void init(){
        img0=getImage(getCodeBase(),"road.gif");
        img1=getImage(getCodeBase(),"car.gif");
}
public void paint(Graphics g){
g.drawImage(img0,0,10,this);
g.drawImage(img1,x,30,this);
g.drawImage(img0,0,60,this);
try{
        Thread.sleep(50);
        x+=5;
        if(x==550){
                x=10;
                Thread.sleep(1500);
        }
}catch(InterruptedException e){}
repaint();
        }
}
```

注： road.gif 为公路边线图，　car.gif 为自行车图片

第 10 章

1. 简述什么是　**HTML?**

HTML （Hyper Text Markup Language ）超文本标识语言，所谓超文本，是指除了文本内容外，还可以表现图形、图像、音频、视频、链接等非文本要素。事实上，　Internet　上众多网页的基础，就是　HTML　语言，特别是在互联网发展的初期，几乎所有的网页都是直接用 HTML　语言来书写的。　HTML　是标签式的脚本语言。

2.　**HTML**　对编辑器有什么要求？

HTML　文档是纯文本类型的，因此任何文本编辑器都可以用来对其进行编辑。

3. **HTML** 文件的结构是怎么样的？

HTML 的基本结构是由文档头、文档体构成的，如下所示：

<HTML>

<HEAD>

头 部 信 息

</HEAD>

<BODY>

文 档 主 体， 正 文 部 分

</BODY>

</HTML>

4. 网页中的表格由哪几部分组成 **?**

表格的基本结构可以通过下述标签来定义：

<table>...</table> 定义表格

<caption>...</caption> 定义标题

<tr> 定义表行

<th> 定义表头

<td> 定义表元（表格的具体数据）

5. **HTML** 与 **DHTML** 关系及区别？

HTML 是 DHTML 的基础，DHTML 是在 HTML 的基础上引入动态脚本语言而来； HTML 主要处理静态的网页内容，而 DHTML 还可以通过脚本编程使得页面元素“动”起来。

6. 尝试设计制作自己的个人主页。

（略）

第 11 章

1. 图形用户界面的设计原则有哪些？

通常，图形用户界面的开发都要遵循一些设计原则，比如：

（1）用户至上的原则。设计界面时一定要充分考虑用户的实际需要，使程序能真正吸引住用户，让用户觉得简单易用。

（2）交互界面要友好。

（3）配色方案要合理。建议用柔和的色调，不用太刺眼的颜色。

2. **AWT** 组件集提供的组件大致可以分为哪几类？各起有什么作用？

AWT 组件大致可以分为以下三类：

（1）容器类组件

（2）布局类组件

（3）普通组件类

容器类组件可以用来容纳其他普通组件或者甚至是容器组件自身，起到组织用户界面的作用；布局类组件本身是非可视组件，但它们却能很好地在容器中布置其他普通可视组件；

AWT 提供了一系列的普通组件以构建用户图形界面，它们主要包括：标签、文本框、文本域、按钮、复选框、单选框、列表框、下拉框、滚动条和菜单等。

3. AWT 提供的布局方式有哪几种？请分别进行简述。

（1）FlowLayout

FlowLayout 是最简单的一种布局方式，被容纳的可视组件从左向右，从上至下依次排列，若一组件在本行放置不下，就自动排到下一行的开始处，该方式为 Panel 类和 Applet 类容器的默认布局方式。

（2）BorderLayout

BorderLayout 布局方式的特点是：将容器划分为“东”“西”“南”“北”“中”五个区，分别为 BorderLayout.EAST 、BorderLayout. WEST 、BorderLayout.SOUTH 、BorderLayout.NORTH 和 BorderLayout.CENTER ，每个区可以摆放一个组件，因此最多可以在 BorderLayout 的容器组件中放置五个子组件，前面已提到过，该布局方式是 Frame 或 Dialog 容器类组件的默认布局方式。

（3）GridLayout

GridLayout 布局将容器划分为行和列的网格，每个网格单元可以放置一个组件，组件通过 add（）方法从上到下，从左至右顺序加入网格各个单元中，因此，在使用这种布局时，用户应首先设计好排列位置，然后再依次调用 add（）方法进行添加。

（4）GridBagLayout

GridBagLayout 是所有 AWT 布局管理方式中最繁的，同时也是功能最强的，这主要是因为它提供了许多可设置参数，使得容器的布局方式可以得到准确的控制，尽管设置步骤相对要复杂得多，但是只要理解了它的基本布局思想，就可以很容易使用 GridBagLayout 来进行界面设计了。

（5）CardLayout

CardLayout 布局将组件（通常是 Panel 类的容器组件）象扑克牌（卡片）一样摞起来，每次只能显示其中的一张，实现分页的效果，每一页中可以有各自的界面，这样就相当于扩展了原本有限的屏幕区域。

4. 简述如何创建 **AWT** 的菜单系统。

AWT 提供的菜单系统类包括： MenuBar、MenuItem、Menu、CheckboxMenuItem 以及 PopupMenu。MenuBar 类对应菜单系统的整体， Menu 类对应菜单系统中的一列菜单（实际上它只是一种特殊的菜单项） ，MenuItem 和 CheckboxMenuItem 类则对应具体的菜单项，其中 CheckboxMenuItem 为带复选框的菜单项，而 PopupMenu 类对应弹出式菜单，它是菜单 Menu 类的子类。菜单系统创建好后，最后必须调用 Frame 类的 setMenuBar（）方法将其加入到框架界面中。

5. 简述 **AWT** 提供的基于事件监听模型的事件处理机制。

基于事件监听模型的事件处理是从一个事件源授权到一个或多个事件监听者， 组件作为事件源可以触发事件，通过 addXXXlistener （）方法向组件注册监听器，一个组件可以注册多个监听器， 如果组件触发了相应类型的事件， 此事件被传送给已注册的监听器， 事件监听器通过调用相应的实现方法来负责处理事件的过程。

6. 列出几个你所熟悉的 **AWT** 事件类，并举例说明什么时候会触发这些事件。

（1）ActionEvent 类：可以是鼠标单击按钮或者菜单，也可以是列表框的某选项被双击或文本框中的回车行为。

（2） KeyEvent 类：当用户按下或释放键时产生该类事件，也称为键盘事件。

(3) MouseEvent 类：当用户按下鼠标、释放鼠标或移动鼠标时会产生鼠标事件。

7. **AWT** 规定的 MouseEvent 类对应哪些监听器接口？这些接口中都定义有哪些抽象方法？

（1） MouseListener

```
public abstract void mouseClicked(MouseEvent mouseevent);
public abstract void mousePressed(MouseEvent mouseevent);
public abstract void mouseReleased(MouseEvent mouseevent);
public abstract void mouseEntered(MouseEvent mouseevent);
public abstract void mouseExited(MouseEvent mouseevent);
```

（2） MouseMotionListener

```
public abstract void mouseDragged(MouseEvent mouseevent);
public abstract void mouseMoved(MouseEvent mouseevent);
```

8. 简述 **AWT** 为何要给事件提供相应的适配器（即 **Adapter** 类）？

Java 规定：实现一个接口时必须对该接口的所有抽象方法进行具体的实现， 哪怕有些抽象方法事件用户根本用不上， 也要将其实现， 比如上例中的 keyPressed（）方法， 为此， Java 提供了一种叫做 Adapter（适配器） 的抽象类来简化事件处理程序的编写。 适配器类很简单，它其实就是一个实现了接口中所有抽象方法的“空”类，本身不提供实际功能，

9. 简述 **AWT** 与 **Swing** 组件集间的区别？

AWT 组件集依赖于特定的平台，而 Swing 组件集独立于运行平台；

Swing 组件集实现了模型与视图和组件相分离；

Swing 组件集提供了比 AWT 更多、功能更强的组件，增加了新的布局管理方式（如 BoxLayout ），同时还设计出了更多的处理事件。

10. 创建一个有文本框和三个按钮的框架窗口程序，同时要求按下不同按钮时，文本框中能显示不同的文字。

```
import java.awt.*;

import java.awt.event.*;

public class ActionEvent1

{

    private static Frame frame;       //定义为静态变量以便     main 使用

    private static Panel myPanel; // 该面板用来放置按钮组件

    private Button button1;              //这里定义按钮组件

    private Button button2;              //以便  addActionListener

    private Button button3;

    private TextField textfield1; //  定义文本框组件

    public ActionEvent1()                  //构造方法 ， 建立图形界面

    {

        //  创建面板容器类组件

        myPanel = new Panel();

        //  创建按钮组件

        button1 = new Button("  按钮  1");

        button2 = new Button("  按钮  2");

        button3 = new Button("  按钮  3");

        textfield1 = new TextField(30);

        MyListener myListener = new MyListener();

         //  建立一个   actionlistener 让两个按钮共享

       button1.addActionListener(myListener);

       button2.addActionListener(myListener);

       button3.addActionListener(myListener);

       textfield1.addActionListener(myListener);

       myPanel.add(button1); //   添加组件到面板容器
```

```
        myPanel.add(button2);
        myPanel.add(button3);
        myPanel.add(textfield1);
}
  //定义行为事件处理内部类     ,它实现   ActionListener   接口
    private class MyListener implements ActionListener
{
  /*
     利用该内部类来监听所有行为事件源产生的事件
*/
      public void actionPerformed(ActionEvent e)
      {
          //  利用  getSource()方法获得组件对象名
          //  也可以利用    getActionCommand() 方法来获得组件标识信息
          //  如   e.getActionCommand().equals("  按钮   1")
          Object obj = e.getSource();
          if (obj == button1)
              textfield1.setText("  按钮   1 被单击   ");
          else if (obj == button2)
              textfield1.setText("  按钮   2 被单击   ");
          else if (obj == button3)
              textfield1.setText(" 按钮   3 被单击   ");
          else
              textfield1.setText("");
      }
    }
    public static void main(String s[])
    {
        ActionEvent1 ae = new ActionEvent1(); //     新建   ActionEvent1   组件
        frame = new Frame("ActionEvent1");//     新建   Frame
```

```
// 处理窗口关闭事件的通常方法      (属于匿名内部类    )
frame.addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e)
{System.exit(0);} });
frame.add(myPanel);
frame.pack();
        frame.setVisible(true);
    }
}
```

11. 创建一个带有多级菜单系统的框架窗口程序，要求每单击一个菜单项，就弹出一个相对应的信息提示框。

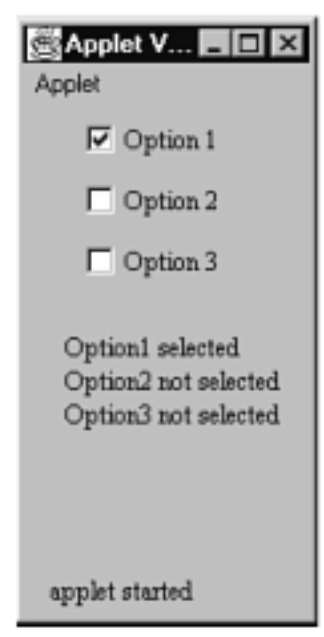
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class DialogFrame extends JFrame
    implements ActionListener
{   public DialogFrame()
    {   setTitle(" 练习  11");
        setSize(300, 300);
        addWindowListener(new WindowAdapter()
        {   public void windowClosing(WindowEvent e)
            {   System.exit(0);
            }
        } );
        JMenuBar mbar = new JMenuBar();
        setJMenuBar(mbar);
        JMenu fileMenu = new JMenu("   文件 ");
        JMenu s = new JMenu(" 特殊功能   ");
        s.add(specials[0]);
```

```
        specials[0].addActionListener(this);
        s.add(specials[1]);
        specials[1].addActionListener(this);
        mbar.add(fileMenu);
        fileMenu.add(s);
        aboutItem = new JMenuItem("  关于 ");
        aboutItem.addActionListener(this);
        fileMenu.add(aboutItem);
        exitItem = new JMenuItem("  退出 ");
        exitItem.addActionListener(this);
        fileMenu.add(exitItem);
    }
    public void actionPerformed(ActionEvent evt)
    {   Object source = evt.getSource();
        if(source == aboutItem)
    {   if (dialog1 == null) // first time
            dialog1 = new AboutDialog(this,"   关于 ");
        dialog1.show();
    }
    else if(source == exitItem)
    {   System.exit(0);
    }
    else if(source == specials[0])
    {   if (dialog2 == null) // first time
            dialog2 = new AboutDialog(this,"   功能  1");
        dialog2.show();
    }
    else if(source == specials[1])
    {   if (dialog3 == null) // first time
            dialog3 = new AboutDialog(this,"   功能  2");
```

```
            dialog3.show();
        }
    }
    private AboutDialog dialog1;
    private AboutDialog dialog2;
    private AboutDialog dialog3;
        private JMenuItem aboutItem;
        private JMenuItem exitItem;
        private JMenuItem[] specials = {
                new JMenuItem(" 功能  1"),
                new JMenuItem(" 功能  2")
            };
}
class AboutDialog extends JDialog
{   public AboutDialog(JFrame parent, String title)
    {   super(parent, title, true);
        JPanel p2 = new JPanel();
        JButton ok = new JButton("Ok");
        p2.add(ok);
        getContentPane().add(p2, "Center");
        ok.addActionListener(new ActionListener()
            {   public void actionPerformed(ActionEvent evt)
                { setVisible(false); }
            } );
            setSize(250, 150);
        }
}
public class DialogTest {
    public static void main(String[] args)
    {   JFrame f = new DialogFrame();
```

```
        f.show();
    }
}
```

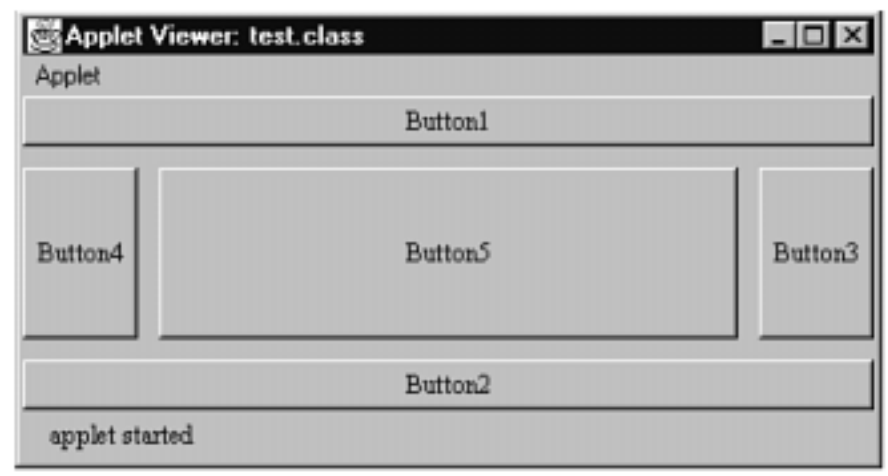
12. 创建 **Applet** 程序，实现以下界面及其功能：




```
import java.awt.*;
import java.applet.*;
public class CheckboxApplet extends Applet
{
    Checkbox checkbox1;
    Checkbox checkbox2;
    Checkbox checkbox3;
    public void init()
    {
        checkbox1 = new Checkbox("Option 1", null, true);
        checkbox2 = new Checkbox("Option 2", null, false);
        checkbox3 = new Checkbox("Option 3", null, false);
        add(checkbox1);
        add(checkbox2);
        add(checkbox3);
    }
    public void paint(Graphics g)
    {
        Font font = g.getFont();
```

```
FontMetrics fontMetrics = g.getFontMetrics(font);
int height = fontMetrics.getHeight();
boolean checked = checkbox1.getState();
if (checked)
        g.drawString("Option1 selected", 20, 120);
else
        g.drawString("Option1 not selected", 20, 120);
checked = checkbox2.getState();
if (checked)
        g.drawString("Option2 selected", 20, 120 + height);
else
        g.drawString("Option2 not selected", 20, 120 + height);
checked = checkbox3.getState();
if (checked)
        g.drawString("Option3 selected", 20, 120 + 2 * height);
        else
                g.drawString("Option3 not selected", 20, 120 + 2 * height);
        }
        public boolean action(Event evt, Object arg)
        {
                repaint();
                return true;
        }
}
```

13. 创建 **Applet** 程序，实现与下图一样的布局界面。



```
import java.awt.*;
import java.applet.*;
public class BorderApplet extends Applet
{
    CardLayout cardLayout;
    Panel panel;
    Button button1, button2, button3;
    public void init()
    {
        panel = new Panel();
        add(panel);
            BorderLayout layout = new BorderLayout(10, 10);
            setLayout(layout);
            Button button1 = new Button("Button1");
            Button button2 = new Button("Button2");
            Button button3 = new Button("Button3");
            Button button4 = new Button("Button4");
            Button button5 = new Button("Button5");
            add("North", button1);
            add("South", button2);
            add("East", button3);
            add("West", button4);
            add("Center", button5);
    }
}
```

14. 请分别用 **AWT** 及 **Swing** 组件来设计实现计算器程序，要求能完成简单四则运算。

下面给出 Swing 版的参考程序， AWT 版的类似。

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

```
class CalculatorPanel extends Jpanel    implements ActionListener
{    public CalculatorPanel()
     {    setLayout(new BorderLayout());
          display = new JTextField("0");
          display.setEditable(false);
          add(display, "North");
          JPanel p = new JPanel();
          p.setLayout(new GridLayout(4, 4));
          String buttons = "789/456*123-0.=+";
          for (int i = 0; i < buttons.length(); i++)
               addButton(p, buttons.substring(i, i + 1));
          add(p, "Center");
     }
private void addButton(Container c, String s)
     {    JButton b = new JButton(s);
          c.add(b);
          b.addActionListener(this);
}
public void actionPerformed(ActionEvent evt)
{    String s = evt.getActionCommand();
     if ('0' <= s.charAt(0) && s.charAt(0) <= '9'
          || s.equals("."))
     {    if (start) display.setText(s);
          else display.setText(display.getText() + s);
          start = false;
}
else
{    if (start)
     {    if (s.equals("-"))
          { display.setText(s); start = false; }
```

```
        else op = s;
    }
    else
    {   double x =
            Double.parseDouble(display.getText());
        calculate(x);
            op = s;
            start = true;
        }
    }
}
public void calculate(double n)
{   if (op.equals("+")) arg += n;
    else if (op.equals("-")) arg -= n;
    else if (op.equals("*")) arg *= n;
        else if (op.equals("/")) arg /= n;
        else if (op.equals("=")) arg = n;
        display.setText("" + arg);
    }
    private JTextField display;
    private double arg = 0;
    private String op = "=";
    private boolean start = true;
}
class CalculatorFrame extends JFrame
{   public CalculatorFrame()
    {   setTitle("Calculator");
        setSize(200, 200);
        addWindowListener(new WindowAdapter()
            {   public void windowClosing(WindowEvent e)
```

```
                {     System.exit(0);
                }
            } );
        Container contentPane = getContentPane();
        contentPane.add(new CalculatorPanel());
    }
}
public class Calculator
{    public static void main(String[] args)
    {    JFrame frame = new CalculatorFrame();
         frame.show();
    }
}
```

第 12 章

1. 以下哪一个为标准输出流类 _____。

A、 DataOutputStream **B、 FilterOutputStream**

C、 PrintStream **D、 BufferedOutputStream**

C

2. 将读取的内容处理后再进行输出，适用下述哪种流 _____。

A、 PipedStream **B、 FilterStream**

C、 FileStream **D、 ObjectStream**

B

3. **DataInput** 和 **DataOutput** 是处理哪一种流的接口 _____。

A、文件流 **B**、字节流

C、字符流 **D**、对象流

B

4. 下面语句正确的是 _____。

A、 RandomAccessFile raf=new RandomAccesssFile(“data.dat ”, ”rw ”);

B、 RandomAccessFile raf=new RandomAccesssFile(new DataInputStream());

C、 RandomAccessFile raf=new RandomAccesssFile(“data.dat ”);

D、 RandomAccessFile raf=new RandomAccesssFile(new File(“data.dat ”));

A

5. 以下不是 **Reader** 基类的直接派生子类的是 _____。

A、 BufferedReader　　**B、 FilterReader**

C、 FileReader　　**D、 PipedReader**

C

6. 测试文件是否存在可以采用如下哪一个方法 _____。

A、 isFile（）　　**B、 isFiles（）**

C、 exist（）　　**D、 exists（）**

D

7. **Java** 中， **InputStream** 和 **OutputStream** 是以 _____为数据读写单位的输入输出流的基类； **Reader** 和 **Writer** 是以 _____为数据读写单位的输入输出流的基类。

字节；字符

8. 以字符方式对文件进行读写可以通过 _____类和 _____类来实现。

FileReader；FileWriter

9. **RandomAccessFile**类所实现的接口有 _____和 _____，调用它的 _____方法可以移动文件位置指针，以实现随机访问。

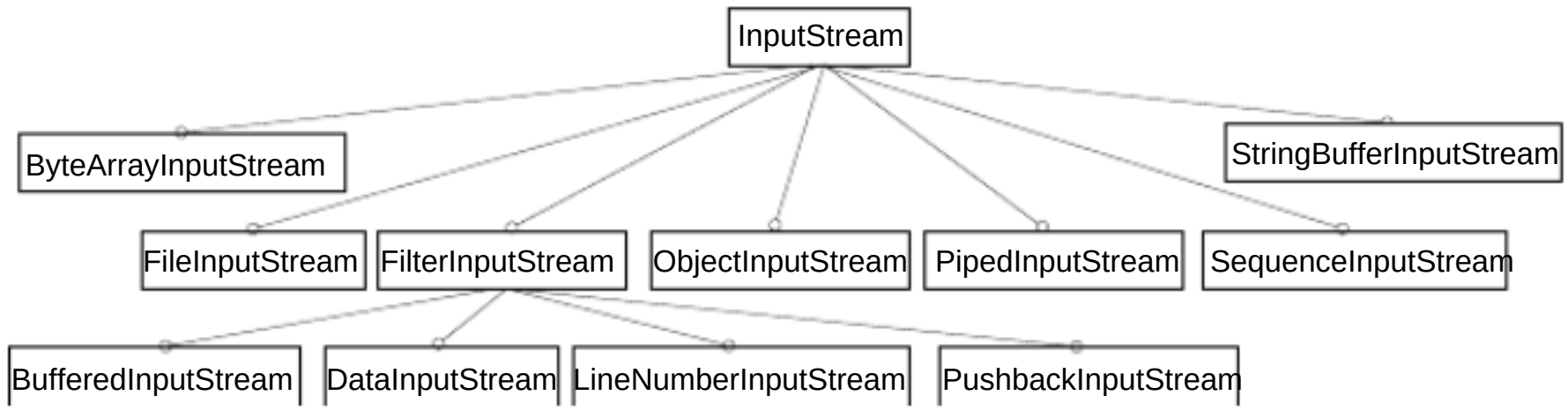
DataInput；DataOutput；seek()

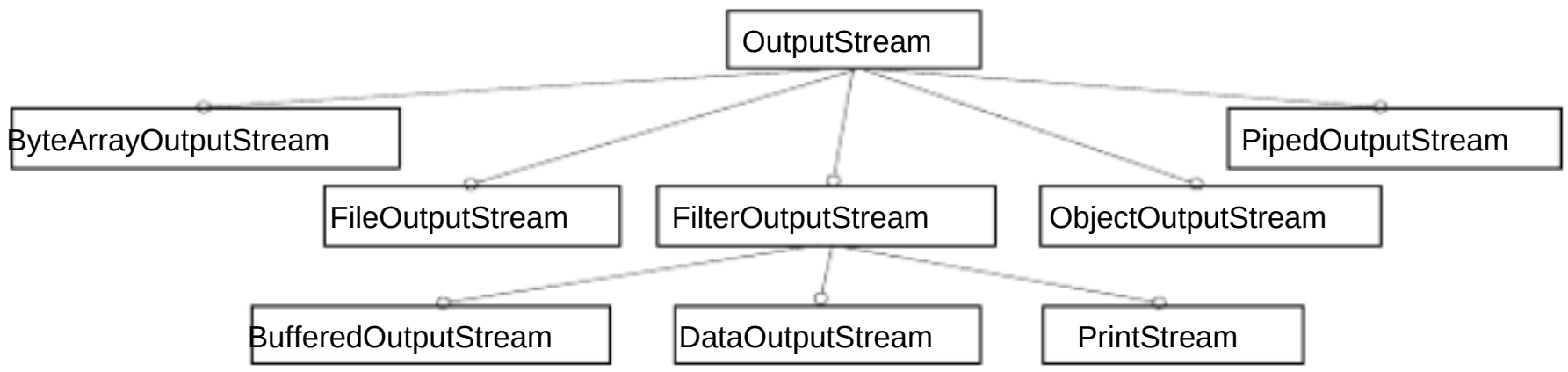
10. 简述 **Java** 中的标准输入输出是怎么实现的。

System 为 Java 自动导入包 java.lang 里的一个类， 它含有三个内建好的静态流对象： err、in 和 out，in 属于 InputStream 对象，而 err 和 out 属于 PrintStream 对象，它们分别用于标准错误输出、标准输入和标准输出， 程序中可以直接使用这三个流对象， 比如调用它们的 println（）或 read（）方法来实现标准输入输出功能，默认情况下，标准输入 in 用于读取键盘输入，而标准输出 out 和标准错误输出 err 用于把数据输出至启动程序运行的终端屏幕上。

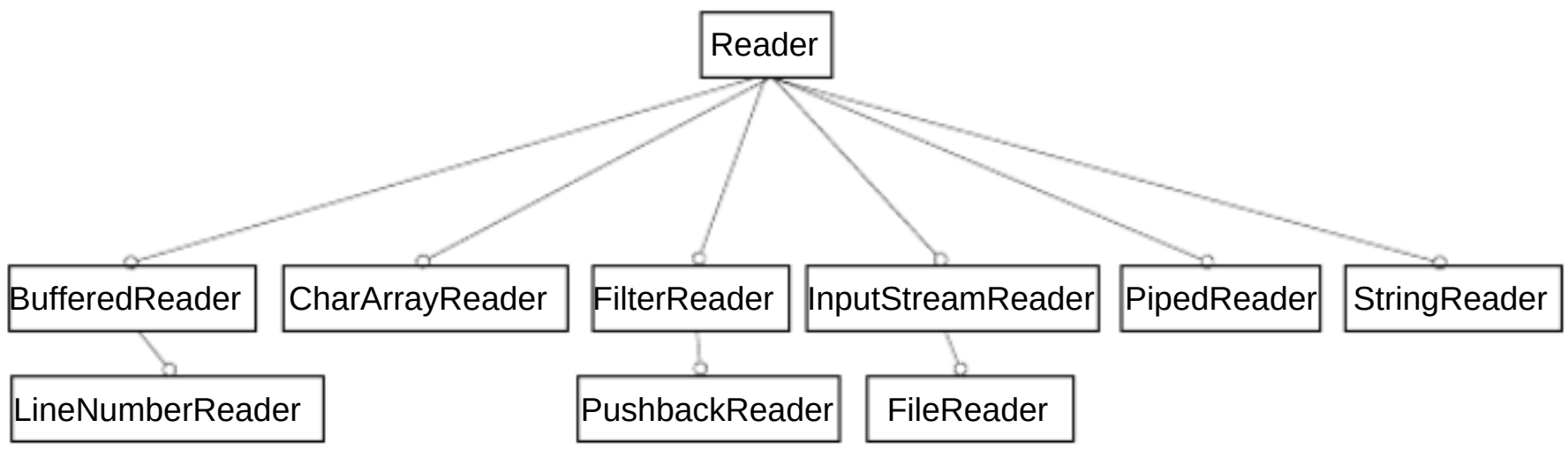
11. 简述 **java.io** 包是如何设计提供字节流和字符流输入输出体系的。

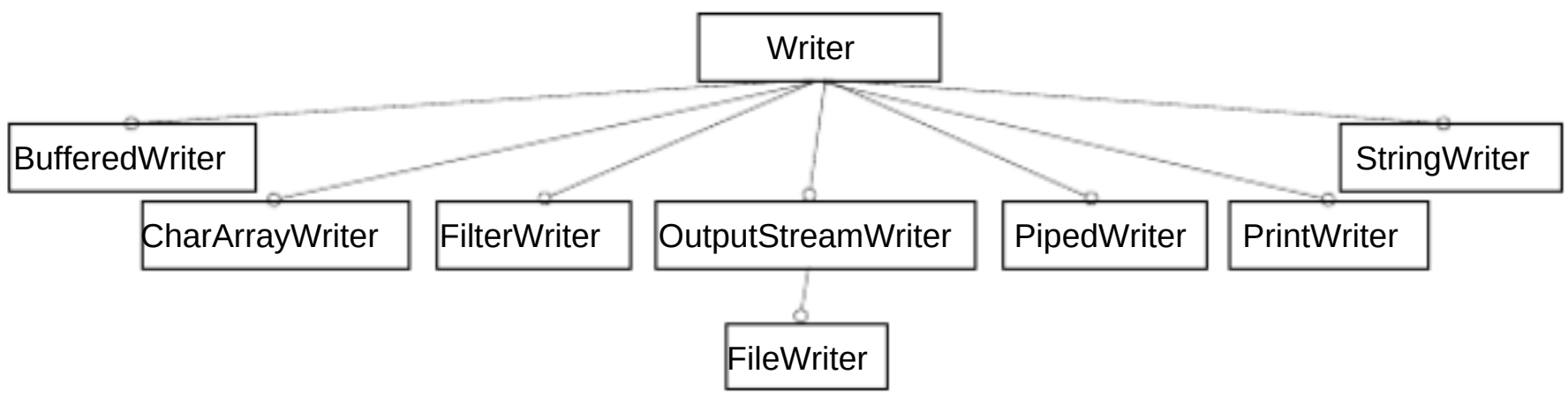
java.io 包里包含了众多的输入输出流类， 其中以字节为处理单位的流称为字节流， 而以字符为处理单位的流称为字符流。 字节输入流的基类为 InputStream，OutputStream 抽象类是所有字节输出流类的基类，它们各自的派生关系如下所示：





而 Java 设计的字符流体系中也有两个基本类： Reader 和 Writer，分别对应字符输入流和字符输出流，它们的派生图如下：





12. 简述 **File** 类的应用，它与 **RandomAccessFile** 类有何区别。

File 类直接处理文件和文件系统本身， 也就是说 File 类并不关心怎样从文件读取数据流或向文件存储数据流，它主要用来描述文件或目录的自身属性。通过创建 File 类对象，我们可以处理和获取与文件相关的信息，比如文件名、相对路径、绝对路径、上级目录、是否存在、是否是目录、可读、可写、上次修改时间和文件长度等。此外，当 File 对象为目录时，还可以列举出它的文件和子目录。

File 类不能进行文件读写操作， 必须通过其他类来提供该功能， RandomAccessFile 类就是其中之一。 RandomAccessFile 类文件存取方式灵活，它支持随机存取（ Random Access）：在文件中可以任意移动读取位置。

13. 编程实现文件内容合并，即将某文件的内容写入到另一个文件的末尾处。

参考程序如下：

```
import java.io.*;
public class Union {
public static void append(String file1, String file2){
    try {
            FileWriter writer = new FileWriter(file1, true);
            FileReader fr = new FileReader(file2);
            BufferedReader bfr = new BufferedReader(fr);
            String str=bfr.readLine();
            while (str!=null)
            {
                writer.write(str);
                str=bfr.readLine();
            }
            writer.close();
```

```
        } catch (IOException e) {
                e.printStackTrace();
    }
}
public static void display(String fileName){
      try
         {
           FileReader fr = new FileReader(fileName);
           BufferedReader bfr = new BufferedReader(fr);
    String str=bfr.readLine();
    while (str!=null)
    {
        System.out.println(str);
        str=bfr.readLine();
    }
  }
catch (IOException ioe)
{
    System.out.println(ioe);
}
catch (Exception e)
{
           System.out.println(e);
       }
}
public static void main(String[] args) {
      String file1 = "F:/me/data1.txt";
      String file2 = "F:/me/data2.txt";
      append(file1, file2);
      display(file1);
   }
}
```

14. 编写一递归程序，列举出某个目录下的所有文件以及所有子目录（包括其下的文件和子目录），要求同时列出它们的一些重要属性。

参考程序如下：

```
import java.io.*;
import java.util.*;
class Lists
{
    static void list(String strPath) throws Exception
    {
      try
      {
          File f=new File(strPath);
```

```
if(f.isDirectory())
{
    File[] fList=f.listFiles();
    for(int j=0;j<fList.length;j++)
    {
        if(fList[j].isDirectory())
      {
          System.out.println(fList[j].getPath());
          list(fList[j].getPath());
      }
}
for(int j=0;j<fList.length;j++)
{
    if(fList[j].isFile())
    {
        System.out.print(fList[j].getPath());
        System.out.print(fList[j].isHidden()? "->    隐藏 " : "-> 没隐藏 ");
        System.out.print(fList[j].canRead() ? "->   可读   " : "-> 不可读   ");
        System.out.print(fList[j].canWrite() ? "->    可写   " : "-> 不可写   ");
        System.out.println("  大小 ->" +fList[j].length() + "    字节 ");
        //System.out.println("  最后修改时间   ->" +new Date(fList[j].lastModified()));
  }
      }
    }
  }
  catch(Exception e)
  {
      System.out.println("Error   :   " + e);
  }

}
  public static void main(String[] args)
{
    String path="F:\\me";
    System.out.println(path);
  try
    {
        list(path);
    }
    catch(Exception e)
    {          }
   }
}
```

15. 尝试自定义两个过滤流子类 （如 **CaseInputStream** 和 **CaseOutputStream**），实现将字母

进行大小写转换功能。例如，键盘标准输入为 **aAbB**，用 **CaseInputStream** 读取后应转换为 **AaBb**，再用 **CaseOutputStream** 进行输出时，又会变为 **aAbB**。

参考程序如下：

```
//CaseInputStream 类
import java.io.*;
public class CaseInputStream extends FilterInputStream {
        public CaseInputStream (InputStream i) {
                super(i);
        }
        // override abstract method: read()
        public int read() throws IOException {
                return Trans(in.read());
        }
        private int Trans(int c) {
                if ( (c >= 'A') && (c <= 'Z') )
                        c=c+32;
                if ( (c >= 'a') && (c <= 'z') )
                        c=c-32;
                return c;
        }
}
// CaseOutputStream 类
import java.io.*;
public class CaseOutputStream extends FilterOutputStream {
        public CaseOutputStream (OutputStream i) {
                super(i);
        }
        // override abstract method: write(int c)
        public void write(int c) throws IOException {
                out.write(Trans(c));
        }
        private int Trans(int c) {
                if ( (c >= 'A') && (c <= 'Z') )
                        c=c+32;
                if ( (c >= 'a') && (c <= 'z') )
                        c=c-32;
                return c;
        }
}
```

16. 编一程序。要求

（**1**）在当前目录下创建文件 **students.dat**。

（**2**）录入一批同学的身份证号、姓名和高考总分到上述文件中。

（**3**）提供查询第ｎ位同学信息的功能。

（**4**）提供删除第ｎ位同学信息的功能。

（**5**）提供随机录入功能，即新录入的同学信息可以插入到第ｎ位同学之后。

参考程序如下：

```
import java.io.*;
import java.util.*;
import myPackage.MyInput;
class Student {
        private StringBuffer id;
    private StringBuffer name;
    private short score;
    public Student(String pid,String n,int p) {
        id = new StringBuffer(pid);
        id.setLength(16);
        name=new StringBuffer(n);
        name.setLength(7);
        score=(short)p;
    }
    public String getId() {
        return id.toString();
    }
    public String getName() {
        return name.toString();
    }
        public short getScore() {
            return score;
        }
        public static int size() {
            return 48;
        }
}
public class Students {
        public static void create() throws IOException
        {
            Student[] students = {new Student("3001","   李白 ", 440),
                                  new Student("3002"," 杜甫 ", 438),
                                  new Student("3003"," 王维 ", 429),
                                  new Student("3004"," 孟浩然  ", 432),
                                  new Student("3005"," 齐白石  ", 418),
                                  new Student("3006"," 张大千  ", 412)};
    File f = new File("students.dat");
    //以读写方式打开     students.dat 文件
    RandomAccessFile raf = new RandomAccessFile(f, "rw");
    //将  students 中的书本信息写入文件
    for(int i = 0; i < students.length; i++) {
        raf.writeChars(students[i].getId());
```

```
            raf.writeChars(students[i].getName());
            raf.writeShort(students[i].getScore());
        }
        raf.close();
    }
    public static void query() throws IOException
    {
        File f = new File("students.dat");
        RandomAccessFile raf = new RandomAccessFile(f, "r");
        System.out.println(" 请输入序号 :");
        //利用自定义类  MyInput 进行数据输入
        int n = MyInput.intData();
        //通过 seek()定位到第 n 位同学的信息起始位置
        raf.seek((n-1) * Student.size());
        //pid 存放读取到的身份证信息
        char[] pid=new char[16];
        char ch;
        for(int i=0;i<16;i++){
            ch = raf.readChar();
            if (ch==0)
                pid[i]='\0';
            else
                pid[i]=ch;
        }
        System.out.print(" 身份证号 :");
        System.out.println(pid);
        //sname 存放读取到的人名
        char[] sname=new char[7];
        for(int i=0;i<7;i++){
            ch = raf.readChar();
            if (ch==0)
                sname[i]='\0';
            else
                sname[i]=ch;
        }
        System.out.print(" 人名 :");
        System.out.println(sname);
        System.out.println(" 高考总分 :" + raf.readShort());
        raf.close();
    }
    public static void delete() throws IOException
    {
    }
    public static void insert() throws IOException
```

```
    {
    }
public static void main(String[] args) throws IOException
{
        System.out.println("---  创建文件并写入信息请按       1 ---");
        System.out.println("---  查询第   n 位同学信息请按       2 ---");
        System.out.println("---  删除第ｎ位同学信息请按       3 ---");
        System.out.println("---  录入新的信息请按             4 ---");
        System.out.println("---  退出请按                     5 ---");
    Label:
    while(true){
       switch(MyInput.intData()){
         case 1: create();    //创建文件并写入信息
                 break;
         case 2: query();      //查询第   n 位同学信息
                 break;
         case 3: delete();    //删除第ｎ位同学信息
                 break;
         case 4: insert();    //录入新的信息
                 break;
         case 5: break Label;
     }
            }
        }
    }
```