第1章 Java 语言面与向对象的程序设计

1. Java语言有哪些主要特点 ?

答：（要点）：

1. 简单易学
2. 面向对象
3. 平台无关性
4. 安全稳定
5. 支持多线程
6. 很好地支持网络编程
7. Java丰富的类库使得 Java可以广泛地应用

2. 简述面向过程问题求解和面向对象问题求解的异同。 试列举出面向对象和面向过程的编程语言各两种。

答：面向过程问题求解，以具体的解题过程为研究和实现的主体，其思维特点更接近于计算机；面向对象的问题求解，则是以 -对象 ‖为主体， -对象 ‖是现实世界的实体或概念在计算机逻辑中的抽象表示，更接近于人的思维特点。

面向过程的编程语言： C，Pascal， Foratn。

面向对象的编程语言： C++， Java， C#。

3. 简述对象、 类和实体及它们之间的相互关系。 尝试从日常接触到的人或物中抽象出对象的概念。

答：面向对象技术中的对象就是现实世界中某个具体的物理实体在计算机逻辑中的映射和体现。类是同种对象的集合与抽象。类是一种抽象的数据类型，它是所有具有一定共性的对象的抽象，而属于类的某一个对象则被称为是类的一个实例，是类的一次实例化的结果。如果类是抽象的概念，如 -电视机 ‖，那么对象就是某一个具体的电视机，如 -我家那台电视机 ‖。

4. 对象有哪些属性 ?什么是状态 ?什么是行为 ?二者之间有何关系 ?设有对象 -学生 ‖，试为这个对象设计状态与行为。

答：对象都具有状态和行为。

对象的状态又称为对象的静态属性，主要指对象内部所包含的各种信息，也就是变量。每个对象个体都具有自己专有的内部变量，这些变量的值标明了对象所处的状态。

行为又称为对象的操作，它主要表述对象的动态属性，操作的作用是设置或改变对象的状态。

学生的状态：姓名、性别、年龄、所在学校、所在系别、通讯地址、电话号码、入学成绩等；

学生的行为：自我介绍、入学注册、选课、参加比赛等。

5. 对象间有哪三种关系 ?对象 -班级 ‖与对象 -学生 ‖是什么关系 ?对象 -学生 ‖与对象 -大学生 ‖是什么关系 ？

答：对象间可能存在的关系有三种：包含、继承和关联。

对象 -班级 ‖与对象 -学生 ‖是包含关系。

对象 -学生 ‖与对象 -大学生 ‖是继承关系。

6. 有人说 -父母 ‖和 -子女 ‖之间是继承的关系。这种说法是否正确 ?为什么？

答： -父母 ‖和 -子女 ‖之间不是面向对象意义上的 -继承 ‖关系。因为这里的继承关系是 -is a‖的关系， -男人 ‖与 -人 ‖之间可以说是继承关系。

7. 面向对象的软件开发包括哪些过程 ?OOA 模型包括哪三个层次 ?OOD 模型在 OOA 模型的基础上引入了哪些工作 ？

答：面向对象的软件开发过程可以大体划分为面向对象的分析 (Object Oriented analysis, OOA)、面向对象的设计 (Object oriented design, OOD)、面向对象的实现 (Object oriented programming, OOP)三个阶段。

面向对象的分析的主要作用是明确用户的需求，并用标准化的面向对象的模型规范地表述这一需求，最后将形成面向对象的分析模型。

面向对象的设计将在 OOA 模型的基础上引入界面管理、任务管理和数据管理三部分的内容。

8. 面向对象的程序设计方法有哪些优点 ?

答：由于对象的概念能够以更接近实际问题的原貌和实质的方式来表述和处理这些问题，所以面向对象的软件开发方法比以往面向过程的方法有更好的灵活性、可重用性和可扩展性，使得上述“分析—设计—实现”的开发过程也更加高效、快捷。

第2章 简单的 Java程序

1. 简述 Java编译和运行的基本方法。

答：编译可以使用 JDK 中的工具 javac.exe。例如：

javac HelloWorldApp.java

运行 Java程序则可以使用 JDK 提供的解释器是 java.exe。例如：

java HelloWorldApp

2. 下载并安装 JDK 软件包， 3. 尝试查看其中的 JDK 文档。

答： Java编程的基本工具包是 JDK （Java Development Kit ）。JDK 是 Sun公司免费提供的开发、运行 Java程序的基本软件，它可以在 Windows 及 Unix 两种平台下使用。常用的版本是 JDK1.2.2， JDK1.3.0，JDK1.4 等。可以从 http://java.sun.com 网站下载较新的版本， 如JDK1.5（也称为 JDK5.0）。

JDK 文档也可以从网上下载。

3. 编写一个 Java Application，利用 JDK 软件包中的工具编译并运行这个程序，在屏幕上输出“Welcome to Java World!”。

答：见程序。

```
public class Ex2_3
{
        public static void main(String[] args)
        {
                System.out.println("Welcom to java vorld");
        }
}
```

4. 编写一个 Java Applet，使之能够在浏览器中显示 “Welcome to Java Applet World!”的字符串信息。

答：见程序。 import java.awt.*;

```
import java.applet.*;
public class Ex2_4 extends Applet { //an applet
        public void paint(Graphics g){
                g.drawString ("Welcome to Java Applet      World!",20,20);
        }
}
```

5. 编写一个 HTML 文件，将上题中生成的 Applet 字节码嵌入其中，并用 WWW 浏览器观看这个 HTML 文件规定的 Web页面。

答：与上一题同。

6. 编写一个程序，能够从键盘上接收两个数字，然后计算这两个数的积。

答：见程序。

```
import java.io.*;
public class Ex2_6
{
	public static void main(String[] args)
	{
		String s = "";
		double c = 0;
		double d = 0;
		try{
			BufferedReader in = new BufferedReader(
				new InputStreamReader( System.in ) );
			System.out.print(" 请输入一个数  : ");
			s = in.readLine();
			c = Double.parseDouble( s );
			System.out.print(" 请输入另一个数  : ");
			s = in.readLine();
			d = Double.parseDouble( s );
		}catch(IOException e){}
		System.out.println(" 这两个数的积为  : " + (c * d) );
	}
}
```

7．编写一个程序，从两个文本框中接收两个数字，然后计算这两个数的积。

答：见程序。

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class    Ex2_7 extends Applet
{
	TextField in1 = new TextField(10);
	TextField in2 = new TextField(10);

	Button btn = new Button("  求两个数的积  ");
	Label out = new Label(" 用于显示结果的标签    ");

	public void init()
	{
		setLayout( new FlowLayout() );
		add( in1 );
		add( in2 );
		add( btn );
		add( out );
		btn.addActionListener( new BtnActionAdapter() );
	}

	class BtnActionAdapter implements ActionListener
	{
		public void actionPerformed( ActionEvent e )
		{
			String s1 = in1.getText();
			double d1 = Double.parseDouble( s1 );
			String s2 = in2.getText();
			double d2 = Double.parseDouble( s2 );
			double result = d1 * d2;
```

```
                out.setText( d1 + "X" + d2 +"=" + result);
            }
        }
    }
```

8. 常用的集成开发工具有哪些？各有什么特点？

答：常用的集成开发工具包括： Borland 公司出品的 Jbuilder, Sun公司出品的 Java Workshop, IBM 公司的 Visual Age for Java , Oracle公司的 Java Develop,等等。另外, Symantec公司的 Visual Caf é也是著名的 Java开发工具。近来, Eclipse也是使用很广的集成开发工具。

第3章 数据运算、流控制、数组

1. 简述 Java程序的构成。如何判断主类 ?下面的程序有几处错误 ?如何改正，这个程序的源代码应该保存成什么名字的文件 ? public class MyJavaClass

```
    {
        public static void main()
        {
            System.out.  printIn("Am I wrong?")  ;
        }
        System.out.  printIn(" 程序结束。 ") ;
    }
```

答：一个复杂的程序可由一个至多个 Java源程序文件构成，每个文件中可以有多个类定义。一般的 Java源程序文件由以下三部分组成：

package 语句； （0句—1句）
import 语句； （0句—多句）
类定义 (1个—多个类定义)

这里所说的主类是指程序中含有 main() 方法的类。

上面的示例程序中的错误主要在于拼写，如：点及分号应该用西文的； printIn 中的大写字母I应改为小写字母 l；另外, main() 方法应该带参数，改为：

```
    public static void main(String [] argvs)
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class  Ex2_7 extends Applet
{
    TextField in1 = new TextField(10);
    TextField in2 = new TextField(10);
    Button btn = new Button("  求两个数的积  ");
    Label out = new Label("  用于显示结果的标签  ");
    public void init()
    {
        setLayout( new FlowLayout() );
        add( in1 );
        add( in2 );
        add( btn );
        add( out );
        btn.addActionListener( new BtnActionAdapter() );
    }
    class BtnActionAdapter implements ActionListener
    {
        public void actionPerformed( ActionEvent e )
```

```
            {
                String s1 = in1.getText();
                double d1 = Double.parseDouble( s1 );
                String s2 = in2.getText();
                double d2 = Double.parseDouble( s2 );
                double result = d1 * d2;
                out.setText( d1 + "X" + d2 +"=" + result);
            }
        }
}
```

2. Java有哪些基本数据类型 ?写出 int 型所能表达的最大、最小数据。

答： Java中定义了 4类/8种基本数据类型：

（1）逻辑型 —— boolean

（2）整数型 —— byte, short, int, long

（3）浮点数型 —— float, double

（4）字符型 —— char

其中整型 int 占 4个字节，其范围为 -2147483648-2147483647 。

3. Java的字符采用何种编码方案 ?有何特点 ?写出五个常见的转义符。

答： char(字符型)是用 Unicode 编码表达的字符， 在内存中占两个字节。 由于 Java的字符类型采用了一种新的国际标准编码方案 —— Unicode 编码， 这样便于东方字符和西方字符处理， 这样与其他语言相比， Java处理多语种的能力大大加强。

常见的转义符，如：

\' 单引号字符

\" 双引号字符

\\ 反斜杠字符

\r 回车

\n 换行

4. Java对标识符命名有什么规定，下面这些标识符哪些是对的 ?哪些是错的 ?错在哪里 ？

(1)MyGame (2)_isHers (3)2JavaProgram (4)Java-Visual-Machine (5)_$abc

答：标识符可以由编程者自由指定，但是需要遵循 — 定的语法规定。标识符要满足如下的规定：

(1)标识符可以由字母、数字和下划线 (_)、美元符号 ($)组合而成。

(2)标识符必须以字母、下划线或美元符号开头，不能以数字开头。

在实际应用标识符时，应该使标识符能一定程度上反映它所表示的变量、常量、对象或类的意义，这样程序的可读性会更好。

题中的标识符中的（ 3）不对，因为用数字开头了， (4)也不对，因为其中用了减号。

5. 什么是常量 ?什么是变量 ?字符变量与字符串常量有何不同 ？

答：常量是在程序运行的整个过程中保持其值不改变的量。变量是在程序的运行过程中数值可变的数据，通常用来记录运算中间结果或保存数据。

字符变量中存放一个字符，字符串常量中存放一串字符。

6. 什么是强制类型转换 ?在什么情况下需要用到强制类型转换 ？

答：强制类型的基本方式是指用以下方式显式地进行数据类型的转换：

（类型）表达式

一般地说，在赋值运算符两侧的类型不一致的情况下，则需要进行自动或强制类型转换。变量从占用内存较少的短数据类型转化成占用内存较多的长数据类型时，可以不做显式的类型转换， Java会自动转换；而将变量从较长的数据类型转换成较短的数据类型时，则必须做强制类型转换。

7. Java有哪些算术运算符、关系运算符、逻辑运算符、位运算符和赋值运算符 ?试列举单

目和三目运算符。

答：在 Java中，按照运算符功能来分，基本的运算符有下面几类：

1．算术运算符 (+，-，*，/，%，++，--)

2．关系运算符 (>，<，>=，<=，==，!=)

3．布尔逻辑运算符 (!，&&，||，&，|)

4．位运算符 (>>，<<，>>>，&，|，^，~)

5．赋值运算符 (=，及其扩展赋值运算符如 +=)

6．条件运算符 (？：)

7．其他 (包括分量运算符．，下标运算符 []，实例运算符 instanceof，内存分配运算符 new，强制类型转换运算符 (类型)，方法调用运算符 () 等)

其中，单目运算符如 -，!，而三目运算符只有一个，即条件运算符（？ ：）。

8．编写一个字符界面的 Java Application 程序，接受用户输入的一个浮点数，把它的整数部分和小数部分分别输出。

答：见程序。

```
/*  原来的程序
    public class MyJavaClass
    {
   public static void main()
    {
    System.out.  println("Am I wrong?")  ;
    }
    System.out.  println(" 程序结束。 ") ;
    }
*/
public class Ex3_1
{
    public static void main(String [] args)
    {
        System.out.println("Am I wrong?");
        System.out.println(" 程序结束。 ");
    }
}
```

9．编写一个字符界面的 Java Application 程序，接受用户输入的 10个整数，比较并输出其中的最大值和最小值。

答：见程序。

```
import java.io.*;
public class Ex3_9
{
    public static void main(String[] args)
    {
        int N = 10;
        int [] a = new int[N];      //声明数组并分配空间

        try{
            BufferedReader in = new BufferedReader(
                new InputStreamReader( System.in ) );
            for( int i=0; i<N; i++ ){
                System.out.print(" 请输入第 " + (i+1) +" 个整数：");
                String s = in.readLine();
                a[i]  = Integer.parseInt( s );
            }
            int max = a[0];
```

```
                int min = a[0];
                for( int i=1; i<N; i++ ){
                    if( max < a[i] ) max = a[i];
                    if( min > a[i] ) min = a[i];
                }
                System.out.println( " 最大数为  : " + max );
                System.out.println( " 最小数为  : " + min );
            }catch(IOException e){}
        }
    }
```

10．编写一个字符界面的　Java Application 程序，接受用户输入的字符，以　　-# 标志输入的结束；比较并输出按字典序最小的字符。

答：见程序。　import java.io.*;

```
public class Ex3_10
{
    public static void main(String[] args)
    {
        char min = '\uffff';

        System.out.print(" 请输入一串字符，以　#结束： ");
        while( true )
        {
            char c = ' ';
            try{
                c = (char) System.in.read();

            }catch(IOException e){}

            if( c == '#' ) break;
            if( min > c ) min = c;
        }
        System.out.println("\n 其中最小的字符是　: " + min );
    }
}
```

11. 结构化程序设计有哪三种基本流程　　?分别对应 Java中的哪些语句　?

答：任何程序都可以且只能由三种基本流程结构构成,　　即顺序结构、 分支结构和循环结构。顺序结构直接书写，分支结构用　　if 及switch 语句书写，循环结构用　 for、 while 及do语句来书写。

12．编写一个 Java程序，接受用户输入的一个　　l-12之间的整数 (如果输入的数据不满足这个条件，则要求用户重新输入　　)，利用 switch 语句输出对应月份的天数。

答：见程序。　import java.io.*;

```
public class Ex3_12
{
    public static void main(String[] args)
    {
        try{
            BufferedReader in = new BufferedReader(
                new InputStreamReader( System.in ) );

            int a = 0;
            do{
```

```
        System.out.print(" 请输一个  1~12的整数 : ");
        String s = in.readLine();
        a = Integer.parseInt( s );
}while( a <1 || a > 12 );

switch( a ){
        case 1: System.out.println(" 一月 "); break;
        case 2: System.out.println(" 二月 "); break;
        case 3: System.out.println(" 三月 "); break;
        case 4: System.out.println(" 四月 "); break;
        case 5: System.out.println(" 五月 "); break;
        case 6: System.out.println(" 六月 "); break;
        case 7: System.out.println(" 七月 "); break;
        case 8: System.out.println(" 八月 "); break;
        case 9: System.out.println(" 九月 "); break;
                case 10: System.out.println(" 十月 "); break;
                case 11: System.out.println(" 十一月  "); break;
                case 12: System.out.println(" 十二月  "); break;
                default: break;
                }
        }catch(IOException e){}
    }
}
```

13. 在一个循环中使用 break, continue和return语句有什么不同的效果 ?

答： break是结束循环； continue是直接进行下一次循环； return则是结束整个函数的调用。

14. 编写图形界面下的 Java Applet 程序，接受用户输入的两个数据为上、下限，然后 10个一行输出上、下限之间的所有素数。

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class  Ex3_14 extends Applet
{
    TextField in1 = new TextField(10);
    TextField in2 = new TextField(10);
    Button btn = new Button("  求两个数之间的素数  ");
TextArea out = new TextArea(10,100); //  用于输出
public void init()
{
    setLayout( new FlowLayout() );
    add( in1 );
    add( in2 );
    add( btn );
    add( out );
    btn.addActionListener( new BtnActionAdapter() );
}

class BtnActionAdapter implements ActionListener
{
    public void actionPerformed( ActionEvent e )
    {
        String s1 = in1.getText();
```

```
int a1 = Integer.parseInt( s1 );
String s2 = in2.getText();
double a2 = Integer.parseInt( s2 );

String result = ""; // 结果
int cnt = 0; // 计算素数的个数
for( int i = a1; i<=a2; i++ ){
        if( isPrime( i ) ){
                result +=     (""+i); //  如果是素数，则加入
                cnt ++;
                if( cnt % 10 == 0 ) result += "\r\n";
                else result += "\t";
        }
}
out.setText( result);
                }
                public boolean isPrime( int a )//  判断一个数是否是素数
                {
                        if( a <= 1 ) return false;
                        for( int i=2; i<a; i++ )
                                if( a % i == 0 ) return false;
                        return true;
                }
        }
}
```

15. 编写程序输出用户指定数据的所有素数因子。

答：见程序。

```
import java.io.*;
public class Ex3_15
{
        public static void main(String[] args)
        {
                try{
                        BufferedReader in = new BufferedReader(
                                new InputStreamReader( System.in ) );

                        System.out.print(" 请输入一个正整数    : ");
                        String s = in.readLine();
                        int a     = Integer.parseInt( s );

                        System.out.println( a + " 的所有素因子包括    : ");
                        for( int i=2; i<=a; i++ ){
                                if( a%i==0 && isPrime(i) ){
                                        System.out.println( i );
                                }
                        }

                }catch(IOException e){}
}

public static boolean isPrime( int a )//  判断一个数是否是素数
{
        if( a <= 1 ) return false;
```

```
            for( int i=2; i<a; i++ )
                if( a % i == 0 ) return false;
            return true;
        }
    }
```

16．什么是数组 ?数组有哪些特点 ?Java中创建数组需要使用哪些步骤 ?如何访问数组的一个元素 ?数组元素的下标与数组的长度有什么关系 ?

答：数组是有序数据的集合，数组中的每个元素具有相同的数据类型，可以用一个统一的数组名和下标来惟一地确定数组中的元素。

数组要进行定义和分配空间后才能使用。

一维数组的定义方式为：

type arrayName[ ]；

或 type [ ] arrayName ；

分配内存空间则要用到运算符 new，其格式如下：

arrayName = new type[arraySize] ；

当定义了一个数组，并用运算符 new为它分配了内存空间后，就可以引用数组中的每一个元素了。数组元素的引用方式为：

arrayName[index]

数组元素的下标从 0开始，一直到数组的长度减 1。

17．数组元素怎样进行缺省的初始化 ?

答：数组元素可以在定义数组的同时进行初始化。例如：

int a[ ] = {1,2,3,4,5};

用逗号 (，)分隔数组的各个元素，系统自动为数组分配一定的空间。

18．编程求一个整数数组的最大值、最小值、平均值和所有数组元素的和。

答：见程序。 import java.io.*;

```
public class Ex3_18
{
    public static void main(String[] args)
    {
        int N = 100;
        int [] a = new int[N];
        for( int i=0; i<N; i++ ) {
            a[i]=(int) (Math.random()*100); //  为了简单，这里赋随机值
        }

        int sum = 0;
        int max = a[0];
        int min = a[0];
        for( int i=0; i<N; i++ ){
            sum += a[i];
            if( max < a[i] ) max = a[i];
            if( min > a[i] ) min = a[i];
        }

        System.out.println( " 最大值： " + max );
        System.out.println( " 最小值： " + min );
        System.out.println( " 总和： " + sum );
        System.out.println( " 平均值： " + ((double)sum/N) );
}
```

}

19．求解 -约瑟夫问题 ‖：12个人排成一圈，从 1号报数，凡是数到 5的人就出队列（出局） ，然后继续报数，试问最后一人出局的是谁。

答： import java.io.*;

```
public class Ex3_19
{
    public static void main(String[] args)
    {
        int N = 12;
        boolean [] a = new boolean[N]; //  队列状态
        for( int i=0; i<N; i++ ) a[i]=true;

        int i =0; //  当前报数的人的下标
        int s = 0; // 当前报的数字
        int cnt = 0; // 已有多少人出局
        while(true){
            if( a[i] ) s++;      //如果在队列中，则报数
            if( s == 5 ){//  如果数到 5
                System.out.println( (i+1) ); //  显示该人的序号
                a[i] = false; // 该人出局
                s = 0; //报的数归 0
                cnt ++;
                if( cnt == N )break; //  全部的人都出局，结束
            }
            i++; // 到下一个人
            if( i==N) i=0; //  因为队伍是一个圆圈
        }
    }
}
```

20．用 -埃氏筛法 ‖求2-100以内的素数。 2-100以内的数，先去掉 2的倍数，再去掉 3的倍数，再去掉 4的倍数，以此类推……最后剩下的就是素数。

答：见程序。 import java.io.*;

```
public class Ex3_20
{
    public static void main(String[] args)
    {
        int N = 100;
        boolean [] a = new boolean[N+1];
        for( int i=0; i<=N; i++ ) a[i]=true;

        for( int i=2; i<=N; i++ ){
            for( int j=2*i; j<=N; j += i ){
                a[j] = false;
            }
        }

        for( int i=2; i<=N; i++ ){
            if( a[i] ) System.out.println( i );
        }
    }
}
```

第4章 类、包和接口

1. 使用抽象和封装有哪些好处 ?

答：抽象可以提供事物的本质特征。封装 (Encapsulation) 是这样一种编程机制，它把代码和其操作的数据捆绑在一起，从而防止了外部对数据和代码的干扰和滥用，保证了数据和代码的安全性。

2. 编写一个 Java程序片断定义一个表示学生的类 student，包括域 -学号 ‖ -班号 ‖ -姓名 ‖ -性别 ‖ -年龄 ‖；方法 -获得学号 ‖ -获得班号 ‖ -获得性别 ‖ -获得年龄 ‖ -修改年龄 ‖ class Ex4_2

```
{
    public static void main(String[] args)
    {
    }
class Student
{
    //"学号 "、"班号 "、" 姓名 "、"性别 "、"年龄 "
    String id;
    String classId;
    String name;
    boolean sex;
    int age;

    public String getId() { return id; }
    public String getClassId() { return classId; }
    public String getName() { return name; }
    public boolean getSex() { return sex; }

    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
}
```

3. 为student类定义构造方法初始化所有的域，增加一个方法 public String toString() 把 Sttldent 类对象的所有域信息组合成一个字符串。编写 Application 程序检验新增的功能。

答：见程序。 class Ex4_3

```
{
    public static void main(String[] args)
    {
        Student s = new Student( "09918076", " 软件专业班 ",
            "李明 ", true, 18 );

        System.out.println( s );
    }
}

class Student
{
    //"学号 "、"班号 "、" 姓名 "、"性别 "、"年龄 "
    String id;
    String classId;
```

```
String name;
boolean sex;
int age;

public String getId() { return id; }
public String getClassId() { return classId; }
public String getName() { return name; }
public boolean getSex() { return sex; }

public int getAge() { return age; }
public void setAge(int age) { this.age = age; }

public Student( String id, String classId, String name, boolean sex, int age )
{
        this.id = id;
        this.classId = classId;
        this.name = name;

        public String toString()
        this.age = age;
        this.sex = sex;
        }
        {
        return name + id + classId +
                + age + "岁," +
                (sex ? "男" : "女" ) ;
        }
}
```

4. 什么是最终类，如何定义最终类 ?试列举最终类的例子。

答：最终类是指类被 final 修饰符所修饰和限定的类，最终类不能被继承，即不可能有子类。如Java中的 String 类就是最终类。

5. 如何定义静态域 ?静态域有什么特点 ?如何访问和修改静态域的数据 ?

答：对域使用 static修饰，则定义了静态域。静态域最本质的特点是：它们是类的域，不属于任何一个类的具体对象实例。它不保存在某个对象实例的内存区间中，而是保存在类的内存区域的公共存储单元。

在Java中，静态域（类变量）可以通过类名直接访问，也可以通过实例对象来访问，两种方法的结果是相同的。

6. 什么是静态初始化器 ?它有什么特点 ? 与构造方法有什么不同 ?

静态初始化器是由关键字 static引导的一对大括号 {} 括起的语句组。它的作用与类的构造方法有些相似，都是用来完成初始化的工作，但是静态初始化器在三点上与构造方法有根本的不同：

(1) 构造方法是对每个新创建的对象初始化，而 (2) 静态初始化器是对类自身进行初始化。

(3) 构造方法是在用 new运算符产生新对象时由系统自动执行；而 (4) 静态初始化器一般不 (5) 能由程序来调用， (6) 它是在所属的类加载入内存时由系统调用执行。

(3)不同于构造方法，静态初始化器不是方法，没有方法名、返回值和参数列表。

(4)同static方法一样，它不能访问实例域和实例方法。

7. 如何定义方法 ?在面向对象程序设计中方法有什么作用 ?

答：方法是类的动态属性，标志了类所具有的功能和操作，用来把类和对象的数据封装在一起。 Java的方法与其他语言中的函数或过程类似，是一段用来完成某种操作的程序片断。方

法由方法头和方法体组成，其一般格式如下：

修饰符 1　修饰符 2 ……　返回值类型　　方法名 (形式参数列表)　throws 异常列表 {

方法体各浯句；

}

8. 什么是抽象方法　?它有何特点　?如何定义抽象方法　?如何使用抽象方法　?

答：被 abstract所修饰的方法叫抽象方法，抽象方法的作用在为所有子类定义一个统一的接口。对抽象方法只需声明，而不需实现，即用分号（；　　）而不是用 {}，格式如下：

abstract returnType abstractMethod([paramlist]);

9. 如何定义静态方法　?静态方法有何特点　?静态方法处理的域有什么要求　?

答：用 static修饰符修饰的方法是仅属于类的静态方法，　　又称为类方法。 与此相对， 不用 static修饰的方法，则为实例方法。类方法的本质是该方法是属于整个类的，不是属于某个实例的。

静态方法中，不能访问实例变量。在类方法中不能使用　　this 或 super。

10. 什么是访问控制符　?有哪些访问控制符　?哪些可以用来修饰类　?哪些用来修饰域和方法　?试述不同访问控制符的作用。

答：访问控制符是指　public/private/protected 等表示访问控制　(Access Control) 的修饰符， 其作用是更好地控制类及其域、方法的存取权限，更好地实现信息的封装与隐藏，

方法的访问控制符包括　private, protected, public 和 -默认访问控制符　‖。

类中限定为 private 的成员（域或方法）只能被这个类本身访问，即私有访问控制。

类中的成员缺省访问控制符时，称为缺省访问控制。缺省访问控制的成员可以被这个类本身和同一个包中的类所访问，即包访问控制。

类中限定为 protected的成员可以被这个类本身，它的子类　　(包括同一个包中以及不同包中的子类)以及同一个包中所有其他的类访问。

类中限定为 public 的成员可以被所有的类访问。

Java中还有一种访问控制符为　private proteced，它限定能被本类及其子类可以访问，而包中的其他非子类的类不能访问。

类的访问控制符或者为　public，或者缺省。（但内部类可以有　private、protected等访问控制符。）

如果类用 public 修饰， 则该类可以被其他类所访问；　若类缺省访问控制符，　则该类只能被同包中的类访问。

11. 修饰符是否可以混合使用　?混合使用时需要注意什么问题　?

答：修饰符可以混合使用，如　public static。但也要注意一些规则，如　final 及 abstract不能修饰同一个方法。

12. 什么是继承 ?什么是父类 ?什么是子类 ?继承的特性给面向对象编程带来什么好处　?什么是单重继承 ?什么是多重继承 ?

答：继承 (Inheritance) 是一个对象获得另一个对象的属性的过程。它的重要性源于它支持按层次分类概念。这与现实世界是一致的，大多数知识因为层次化分类而变得容易掌握　(即从上至下)。

在类的层次关系中，被继承的类称为父类，继承其他类的类称为子类。

使用继承，一个对象可以从它父类继承所有的通用属性，而只需定义它特有的属性。　所以，正是继承机制可以使一个对象成为一个更通用类的一个特例成为可能。

一个类只有一个直接父类，称为单重继承；一个类有多个直接父类，则是多重继承。　Java

语言中采用单重继承。

13. 如何定义继承关系？为学生类派生出小学生、中学生、大学生、研究生四个类，其中大学生类再派生出一年级学生、二年级学生、三年级学生、四年级学生四个子类，研究生类再派生出硕士生'和博士生'两个子类。

答：在 Java中，通过 extends关键字来定义继承关系。

具体示例见源程序。

```
class Ex4_13
{
    public static void main(String[] args)
    {
    }

class Student{}
class Pupil extends Student{}
class MiddleSchollStudent extends Student{}
class UnderGraduateStudent extends Student{}
class Graduate extends Student{}

class Freshman extends UnderGraduateStudent{}

class Master extends Graduate{}
class PhD   extends Graduate{}
{
};
}
```

14. 子类的域和方法的数目一定大于等于父类的域和方法的数目，这种说法是否正确?为什么？

答：如果考虑到可以子类既可以继承父类的方法，而不能继承私有的方法，那么这种说法是不正确的。

15. 什么是域的隐藏？

答：在子类中定义与父类同名的域，称为域的隐藏。

16. 什么是方法的覆盖?方法的覆盖与域的隐藏有何不同?与方法的重载有何不同？

答：子类重新定义与父类同名的方法，称为对父类方法的覆盖(Overriding)。子类在重新定义父类已有的方法时，应保持与父类完全相同的方法头声明，即应与父类有完全相同的方法名、返回值和参数类型列表。

方法的重载是指定义几个同名的方法，但它们的参数类型列表不同。

17. 解释 this和super的意义和作用。

答： this表示这个对象本身。详细地说，在普通方法中，this表示调用这个方法的对象；在构造方法中，this表示所新创建的对象。

super指父类。在子类继承父类时，为了明确地指明父类的域和方法，就要用关键字super。

18. 父类对象与子类对象相互转化的条件是什么?如何实现它们的相互转化？

答：类似于基本数据类型数据之间的强制类型转换，存在继承关系的父类对象和子类对象之间也可以在一定条件下相互转换。父类对象和子类对象的转化需要注意如下原则：

(1)子类对象可以被视为是其父类的一个对象。如一个Student对象也是一个 Person对象。

(2)父类对象不能被当作是其某一个子类的对象。

(3)如果一个方法的形式参数定义的是父类对象，那么调用这个方法时，可以使用子类对象作为实际参数。

(4)如果父类对象引用指向的实际是一个子类对象，那么这个父类对象的引用可以用强制类型转换转化成子类对象的引用。

19．构造方法是否可以被继承　?是否可以被重载　?试举例。
答：构造方法不可以被继承，但是可以被子类的构造方法所调用。
构造方法可以被重载。如：

```
Student(String name ){}
Student(String name, int age    ){}
```

20．什么是包　?它的作用是什么　?
答：包是一种松散的类的集合，包的作用在于提供了一种命名机制和可见性限制机制。

21．如何创建包　?在什么情况下需要在程序里创建包　　?
答： package语句作为 Java源文件的第一条语句，指明该文件中定义的类所在的包。它的格式为：

```
package pkg1[.pkg2[.pkg3          … ]];
```

由于 Java编译器为每个类生成一个字节码文件，且文件名与　　public 的类名相同，因此同名的类有可能发生冲突。这时就需要创建包。

22．如何引用包中的某个类　　?如何引用整个包　?如果编写　Java　Applet 程序时想把整个 java．applet包都加载，则应该怎么做　?
答：为了能使用　Java中已提供的类，我们需要用　import 语句来引入所需要的类。　import 语句的格式为：

```
import package1[.package2          … ]. (classname |*);
```

其中 package1[.package2…]表明包的层次，与　package语句相同，它对应于文件目录，　　classname 则指明所要引入的类，如果要从一个包中引入多个类，则可以用星号　　(*) 来代替。例如：

```
import  java.awt.*;
import  java.util.Date;
```

如果编写 Java Applet程序时想把整个　java．applet包都加载，可以使用：

```
import  java.applet.*;
```

23．CLASSPATH 是有关什么的环境变量　?它如何影响程序的运行　?如何设置和修改这个环境变量？
答：在编译和运行程序中，经常要用到多个包，怎样指明这些包的根目录呢？简单地说，包层次的根目录是由环境变量　CLASSPATH 来确定的。具体操作有两种方法：
一是在 java及javac命令行中，用　-classpath选项来指明：
如：

```
java      –classpath  d:\tang\ch04;c:\java\classes;.  pk.TestPkg
```

二是设定 classpath环境变量，用命令行设定环境变量，如：

```
Set CLASSPATH= d:\tang\ch04;c:\java\classes;.
```

在Windows 中还可以按第　2章中的办法设定环境变量。

24．什么是接口　?为什么要定义接口　?接口与类有何异同　?如何定义接口　?使用什么关键字　?
答：Java中的接口　(interface) 在语法上有些相似于类　(class), 它定义了若干个抽象方法和常量，形成一个属性集合，该属性集合通常对应了某一组功能，其主要作用是可以帮助实现类似于类的多重继承的功能。
接口中只能有方法名及常数名，不能像类那样有方法的实现体。
定义接口，使用关键字　interface。
具体方法是：

```
 [public]interface  接口名 [extends父接口名列表  ]
{    //接口体
```

```
//常量域声明
[public][static][final]   域类型域名＝常量值；
//抽象方法声明
[public][abstract]  返回值方法名  (参数列表 )[throw 异常列表 ]；
}
```

25．一个类如何实现接口　?实现某接口的类是否一定要重载该接口中的所有抽象方法　　?

答：一个类要实现接口时，在类的声明部分，用　　implements 关键字声明该类将要实观哪些接口。

如果实现某接口的类不是　abstract的抽象类，则在类的定义部分必须实观指定接口的所有抽象方法。如果其直接或间接父类中实现了接口，父类所实现的接口中的所有抽象方法都必须有实在的方法体。也就是说，非抽象类中不能存在抽象方法。

第5章　深入理解 Java语言

1．什么是多态 ?面向对象程序设计为什么要引人多态的特性　　?使用多态有什么优点　?

答：多态性 (Polymorphism 来自希腊语，　意思是多种形态)是指允许一个接口访问动作的通用类的性质。一般地，多态性的概念常被解释为　　一个接口，多种方法　‖。这意味着可以为一组相关活动设计一个通用接口。多态性允许用相同接口规定一个通用类来减轻问题的复杂度。选择适当的动作 (方法)适应不同环境的工作则留给编译器去做。作为编程者，无需手工去做这些事情，只需利用通用接口即可。

多态的特点大大提高了程序的抽象程度和简洁性，更重要的是，它最大限度地降低了类和程序模块之间的耦合性，提高了类模块的封闭性，使得它们不需了解对方的具体细节，就可以很好地共同工作。这个优点对于程序的设计、开发和维护都有很大的好处。

2．虚方法调用有什么重要作用？具有什么修饰符的方法不能够使用虚方法调用？

答：使用虚方法调用的作用在于：　在使用上溯造型的情况下，子类对象可以当做父类对象，对于重载或继承的方法，　Java运行时系统根据调用该方法的实例的类型来决定选择哪个方法调用。对子类的一个实例，如果子类重载了父类的方法，则运行时系统调用子类的方法，如果子类继承了父类的方法　(未重载)，则运行时系统调用父类的方法。

使用 static、 private 或 final 所修饰的方法不能够使用虚方法调用。

3．用默认构建器（空自变量列表）创建两个类：　　A和B，令它们自己声明自己。从　A继承一个名为 C的新类，并在　C内创建一个成员　B。不要为 C创建一个构建器。创建类　C的一个对象，并观察结果。

答：具体程序见源文件。

程序的运行结果可以看到　A 和B的构造方法都被调用了。　public class Ex5_3

```
{
    public static void main( String[] args )
    {
        C c = new C();
    }
}
class A
{
    public A(){ System.out.println("A"); }
};
class B
{
    public B(){ System.out.println("B"); }
```

```
}
class C extends A
{
        B b = new B();
}
```

4. 创建 Rodent（啮齿动物） :Mouse（老鼠） ,Gerbil （鼹鼠） ,Hamster（大颊鼠）等的一个继承分级结构。在基础类中，提供适用于所有 Rodent的方法，并在派生类中覆盖它们，从而根据不同类型的 Rodent采取不同的行动。创建一个 Rodent数组，在其中填充不同类型的 Rodent，然后调用自己的基础类方法，看看会有什么情况发生。

答：具体程序见源文件。

可以看出虚方法调用带来的多态效果。

```
public class Ex5_4
{
        public static void main( String[] args )
        {
                Rodent [] animals = new Rodent[4];
                animals[0] = new Rodent();
                animals[1] = new Mouse();
                animals[2] = new Gerbil();
                animals[3] = new Hamster();

                for( int i=0; i<animals.length; i++ ){
                        animals[i].eat();
                }
        }
}
class Rodent
{
        public void eat(){ System.out.println("Rodent"); }
}
class Mouse extends Rodent
{
        public void eat(){ System.out.println("Mouse"); }
}
class Gerbil extends Mouse
{
        public void eat(){ System.out.println("Gerbil"); }
}
class Hamster extends Gerbil
{
        public void eat(){ System.out.println("Hamster"); }
}
```

5. Java中怎样清除对象？能否控制 Java中垃圾回收的时间？

答： Java中，无用的对象由系统自动进行清除和内存回收的过程，编程者可以不关心如何回收以及何时回收对象。

对象的回收是由 Java虚拟机的垃圾回收线程来完成的。编程者不能完全控制垃圾回收的时间。

6. 内部类与外部类的使用有何不同？

答：在封装它的类的内部使用内部类，与普通类的使用方式相同；在其他地方使用内部类时，类名前要冠以其外部类的名字才能使用，在用 new创建内部类时，也要在 new前面冠以对象变量。

7. 怎样使用匿名类的对象？

答：在类及其方法中，可以定义一种匿名类，匿名类有以下几个特点：

(1) 这种类不 (2) 取名 (3) 字，而 (4) 直接用其父类的名 (5) 字或者它所实现的接口的名 (6) 字。

(7) 类的定义与创建该类的一个实例同 (8) 时进行， (9) 即类的定义前面有一个 new。也不 (10) 使用关键词 class，同 (11) 时也带上（）表示创建对象。也就是说， (12) 匿 名 (13) 类的定义方法是：

new 类名或接口名 （） { …… }

(14) 类名 (15) 前面不 (16) 能有修饰符。

(17) 类中不 (18) 能定义构造方法， (19) 因为它没有名 (20) 字。在构造对象时， (21) 直接使用父类的构造方法。如果实现接口， (22) 则接口名 (23) 后的圆括号中不 (24) 能带参数。

8. 方法中定义的内部类是否可以存取方法中的局部变量？

答：方法中的内部类，不能访问该方法的局部变量，除非是 final 的局部变量或 final 的参变量。

第6章 异常处理

1. 异常可以分成几类？

答：Java的异常类都是 java.lang.Trowable 的子类。 它派生了两个子类： Exception 和 Error。其中Error 类由系统保留，而 Exception 类则供应用程序使用。

2. 用 main() 创建一个类，令其抛出 try 块内的 Exception 类的一个对象。为 Exception 的构建器赋予一个字串参数。在 catch从句内捕获异常，并打印出字串参数。添加一个 finally 从句，并打印一条消息。

```
public class Ex6_2
{
    public static void main( String[] args )
    {
        try{
            int i = Integer.parseInt( "23.4" );
        }
        catch( Exception e ){
            System.out.println( e );
        }
        finally {
            System.out.println("finished");
        }
    }
};
```

答：见程序。

3. 用 extends关键字创建自己的异常类。为这个类写一个构建器，令其采用 String 参数，并随同 String句柄把它保存到对象内。 写一个方法， 令其打印出保存下来的 String。创建一个 try-catch 从句，练习实际操作新异常。

答：见程序。

```
public class Ex6_3
{
    public static void main( String[] args )
    {
```

```
            try{
                throw new Ex63Exception("  测试 ");
            }
            catch( Ex63Exception e ){
                e.show();
            }
            finally {
                System.out.println("finished");
            }
        }
};

class Ex63Exception extends Exception
{
        String msg;
        public Ex63Exception( String msg ) {
            super( msg );
            this.msg = msg;
        }
        public void show(){
            System.out.println( msg );
        }
    };
```

4. 写一个类，并在一个方法抛出一个异常。试着在没有异常规范的前提下编译它，观察编译器会报告什么。 接着添加适当的异常规范。 在一个 try-catch从句中尝试自己的类以及它的异常。

答：见程序。

```
public class Ex6_4
{
        public static void main( String[] args )
        {
            char c;
            try{
                c = (char)System.in.read();
            }
            // /*  如果不用 catch，编译不会通过
            catch( java.io.IOException e ){
                System.out.println( e );
            }
            // */
        }
};
```

第7章 工具类及常用算法

1. 在所有的 Java系统类中， Object类有什么特殊之处 ?它在什么情况下使用 ？

答： Object类是 Java程序中所有类的直接或间接父类，也是类库中所有类的父类。正因为Object类是所有 Java类的父类， 而且可以和任意类型的对象匹配， 所以在有些场合可以使用它作为形式参数的类型。使用 Object类可以使得该方法的实际参数为任意类型的对象，从而扩大了方法的适用范围。

2. 数据类型包装类与基本数据类型有什么关系 ？

答： Java的基本数据类型用于定义简单的变量和属性将十分方便，但为了与面向对象的环境一致， Java中提供了基本数据类型的包装类（ wrapper）,它们是这些基本类型的面向对象的代表。与8种基本数据类型相对应， 基本数据类型的包装类也有 8种，分别是：Character, Byte， Short，Integer， Long， Float， Double， Boolean。

3. Math类用来实现什么功能 ?设x，y是整型变量， d是双精度型变量， 试书写表达式完成下面的操作：

(1)求x的y次方；
(2)求x和y的最小值；
(3)求d取整后的结果；
(4)求d的四舍五入后的结果；
(5)求atan(d)的数值。

答：相应的表达式为：

(1) Math.pow(x, y)
(2) Math.min(x, y)
(3) Math.floor(d)
(4) Math.rint(d)
(5) Math.atan(d)

4. Math． random()方法用来实现什么功能 ?下面的语句起到什么作用 ？

(int)(Math ． random()*6)+1

答： Math.random() 用于产生随机数（ 0到1之间，包含 0，但不包含 1）。

上面的表达式表示 1到6之间的一个随机整数（包含 1及6）。

5. 编程生成 100个I～6之间的随机数，统计 1—6之间的每个数出现的概率；修改程 序，使之生成 1 000个随机数并统计概率；比较不同的结果并给出结论。

答：见程序。 class Ex7_5

```
{
    public static void main(String[] args)
    {
        statistics( 100 );
        statistics(1000 );
        //结论：次数越大， 6个数出现的机率越接近
    }

public static void statistics( int times )
{
    int [] cnt = new int[6];
    for( int i=0; i<times; i++ )
    {
        int r = rnd( 6 );     //得到一个随机数
        cnt[ r-1 ] ++;          //计数
    }
    System.out.println("1-6 出现的次数分别为 ");
    for( int i=0; i<6; i++ )
    {
        System.out.print( cnt[i] + " " );
    }
    System.out.println();
}
```

```
        public static int rnd( int max )
        {
                return (int)(Math.random() * max ) + 1;
        }
}
```

结论是次数越多越均匀。

6. 什么是字符串 ?Java中的字符串分为哪两类 ?

答：字符串是是字符的序列，在 Java中，字符串，无论是常量还是变量，都是用类的对象来实现的。

字符串可以分为两大类，一类是创建之后不会再做修改和变动的字符串常量；另一类是创建之后允许再做更改和变化的字符串。前者是 String类，后者是 StringBuffer 类。

7. 编写 Applet 程序，接受用户输入的一个字符串和一个字符，把字符串中所有指定的字符删除后输出。

答：

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

public class    Ex7_7 extends Applet
{
        TextField in1 = new TextField(10);
        TextField in2 = new TextField(10);

        Button btn = new Button("  删除字符串的指定字符    ");
        Label out = new Label("  用于显示结果的标签    ");

        public void main int()
        {
                setLayout( new FlowLayout() );
                add( in1 );
                add( in2 );
                add( btn );
                add( out );
        btn.addActionListener( new BtnActionAdapter() );
}

class BtnActionAdapter implements ActionListener
{
        public void actionPerformed( ActionEvent e )
        {
                String s1 = in1.getText();
                String s2 = in2.getText();
                char c2 = s2.charAt(0);

                StringBuffer sb = new StringBuffer();
                for( int i=0; i<s1.length(); i++ ){
                        char c1 = s1.charAt(i);
                        if( c1 != c2 ) {
                                sb.append( c1 );
```

```
                    }
                }

                out.setText( "结果为 " + sb );
            }
        }

}
```

8. 编程判断一个字符串是否是回文。

答： import java.io.*;

```
public class Ex7_8
{
    public static void main(String[] args)
    {
        try{
            BufferedReader in = new BufferedReader(
                new InputStreamReader( System.in ) );
            System.out.print(" 请输入一串字符   : ");
            String s = in.readLine();

            boolean flg = true;
            int len = s.length();
            for( int i=0; i<len/2; i++ ){
                if( s.charAt(i) != s.charAt(len-1-i) ){
                    flg = false;
                    break;
                }
            }
            System.out.println( flg ? "  是回文 " : " 不是回文  " );
        }catch(IOException e){}
    }
}
```

9. String 类的 concat()方法与 StringBuffer 类的 append()方法都可以连接两个字符串，它们之间有何不同 ？

答： String 的 concat()方法不改变原字符串本身，而是产生一个新的字符串。

StringBuffer 的 append()方法则改变其中的字符串内容，而不产生一个新的对象。

10. 什么是递归方法 ?递归方法有哪两个基本要素 ?编写一个递归程序求一个一维数组所有元素的乘积。

答：简单地说，递归方法就是直接或间接调用自己的方法。递归方法有两个要素：一个是求得范围缩小的同性质问题的结果；另一是利用这个已得到的结果和一个简单的操作求得问题的最后解答。 class Ex7_10

```
{

    public static double multi( double [] a )
    {
        return mul( a,    0 ); // 从第 0个元素开始乘
```

```
        }
        //下面的 mul 是一个辅助函数，它表示从第      p个元素开始乘
        public static double mul( double [] a, int p )
        {
                if( p >= a.length ) return 1;
                else return a[p] * mul(a, p+1 ); //   递归调用
        }

        public static void main( String [] args )
        {
                double [] a = { 1, 6, 7, 9 };
                System.out.println( multi(a) );
        }
}
```

具体程序见源文件。

11．你了解几种排序算法　?它们各自有什么优缺点　?分别适合在什么情况下使用　?

答：常见的排序算法有：冒泡排序、选择排序、快速排序等。前二者较简单，适合于元素个数较少的场合，而快速排序法则速度较快，适合于各种场合。

12．向量与数组有何不同　?它们分别适合于什么场合　?

答：向量的元素个数可以动态地增减，而数组则不行。前者适合于元素个数动态改变的场合，后者则适合于元素个数不变的场合。

13． Java中有几种常用的集合类及其区别如何？怎样获取集合中的各个元素。

答：常用的集合有向量、堆栈、队列等。它们可以使用枚举等方法来获集合中的元素，有的还可使用 elemnetAt() 等方法来获取其中的元素。

14．队列和堆栈各有什么特点？

答：队列和堆栈都是线性数据结构，队列遵循　先进先出　(FIFO) 原则，堆栈遵循　后进先出 (LIFO) 原则。

15．求解 "鸡兔同笼问题"：鸡和兔在一个笼里，共有腿　100条，头 40个，问鸡兔各有几只。

答： //Ex 7-15:　鸡兔同笼

```
class   Ex7_15
{
        public static void main(String[] args)
        {
                int chicken, rabbit;
                for( chicken=1; chicken<=40; chicken++ )
                {
                        rabbit = 40-chicken; // 共40头
                        if( rabbit *4 + chicken *2 == 100 )//    共100支脚
                        {
                                System.out.println(" 鸡有 " + chicken + " ， 兔有 " + rabbit);
                        }
                }
        }
}
```

16. 求解“百鸡问题”。已知公鸡每只 3元，母鸡每只 5元，每 3只小鸡 1元。用 100元钱买 100只鸡，问每种鸡应各买多少。

答：见程序。 class Ex7_16

```
{
    public static void main(String[] args)
    {
        int a, b, c; // 三种鸡的只数
        for( a=1; a<=100; a++ ){
            for( b=1; b<100-a; b++ ){
                c = 100 - a - b;
                if( c>0 && c%3==0 && 3*a+5*b+c/3==100 ){
                    System.out.println(" 找到解 :"
                        + a +","+ b +","+ c );
                }
            }
        }
    }
}
```

17. 求四位的水仙花数。即满足这样条样的四位数：各位数字的 4次方和等于该数自身。

答：见程序。 class Ex7_17

```
{
    public static void main(String[] args)
    {
        int a, b, c, d; // 各位数
        for( a=1; a<=9; a++ ){
            for( b=0; b<=9; b++ ){
                for( c=0; c<=9; c++ ){
                    for( d=0; d<=9; d++ ){
                        int s = 1000*a+100*b+10*c+d;
                        int q = a*a*a*a+b*b*b*b+c*c*c*c+d*d*d*d;
                        if( s == q ) System.out.println( s );
                    }
                }
            }
        }
    }
}
```

18. 求 1000以内的“相亲数”。所谓相亲数是指这样的一对数：甲数的约数之和等于乙数，而乙数的约数之和等于甲数。

答：见程序。 class Ex7_18

```
{
    public static void main(String[] args)
    {
        for( int a=1; a<=10000; a++ ){
            int b= sumfactor(a);
            if( b>a && b<=10000 && sumfactor(b)==a ){
                System.out.println( a +"," + b);
```

```
                }
            }
        }
        public static int sumfactor( int a )
        {
            int s = 0;
            for( int i=1; i<a; i++ )
                if( a % i == 0 ) s += i;
            return s;
        }
    }
```

19. 一哥德巴赫猜想 指出，每个大于 6的偶数， 都可以表示为两个素数的和。 试用程序将 6-100 内的所有偶数都表示为两个素数的和。

答：见程序。 //Ex 7-19: 哥德巴赫猜想

```
class  Ex7_19
{
    public static void main(String[] args)
    {
        for( int a = 6; a<=100; a +=2 ) //              对每个偶数进行检验
        {
            for(int i=2; i<a; i++ ) //                  试验每种可能的加法情况
            {
                if( isPrime( i ) && isPrime( a-i ) )//                  两个数都是质数
                {
                    System.out.println(a + "=" + i + "+" + (a-i));
                    break;
                }
            }
        }
    }
    public static boolean isPrime( int m ) //                  判断是否是质数
    {
        for(int i=2; i<m-1; i++ )
        {
            if( m % i == 0 ) return false; //                  如果能除尽，则不是质数
        }
        return true;//        都不能除尽，表明它是质数
    }
}
```

20．菲波那契 (Fibonacci) 数列的第一项是 0，第二项是 I，以后各项都是前两项的和，试用递归算法和非递归算法各编写一个程序，求菲波那契数列第 N项的值。

答：见程序。 public class Ex7_20

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

```
{
        System.out.println(" 非递归方法求得第    5项为：  "
                + fib1( 5) );
        System.out.println(" 递归方法求得第    5项为：  "
                + fib( 5) );
}
static int fib1( int n ){
        if( n==0 || n==1) return 1;
        int a=1, b=1, c=2; //a,b 表示前两项,     c表示当前项
        for( int i=2; i<=n; i++ ){
                c = a+b;
                a = b;
                b = c;
        }
        return c;
        }
        static int fib( int n ){
                if( n==0 || n==1) return 1;
                else return fib(n-1) + fib(n-2);
        }
}
```

21．用迭代法编写程序用于求解立方根。

答：见程序。注意到　x^3-a=0的牛顿迭代公式为　x-(x^3-a)/(3 x^2)

```
public class Ex7_21
{
        public static void main(String args[]){
                System.out.println( trt( 2.0 ) );
                System.out.println( Math.pow(2.0, 1.0/3.0) );
        }

        static double trt( double a ){
                double x=1.0;
                do{
                        x = x - (x*x*x - a)/(3*x*x);
                }while( Math.abs(x*x*x-a)/a > 1e-6 );
                return x;
        }
}
```

。

22．用迭代法编写程序用于求解以下方程：

x^2+sin x-1.0=0　在-1附近的一个根

答：见程序。注意到该方程的牛顿迭代公式为　　x-(x^2+sinx-1.0)/(2 x+cosx)。

```
public class Ex7_22
{
        public static void main(String args[]){
                double x = -1.0;
                int n =0;
                while(true){
                        double xnew = x-( x*x+Math.sin(x)-1.0)/(2*x+Math.cos(x));
```

```
                if( Math.abs(xnew - x) < 1e-6 ) break;
                System.out.println( x );
                x = xnew;
                //if( n++ > 10 )break;

            }
            System.out.println( x );
        }
}
```

23. 作出对应于不同　c值的 Julia集的图：

c=-1+0.005i

c=-0.2+0.75i

c=0.25+0.52i

c=0.5+0.55i

答：见程序。

```
import java.awt.*;
class Ex7_23
{
    public static void main( String []args ){
        Frame frm = new Frame("Julia");
        frm.setSize( 300, 300 );
        frm.setVisible(true);
        Ex7_23 p = new Ex7_23( frm );
        p.drawJulia();
    }

    private Frame frm;
    private Graphics graphics;
    private int width;
    private int height;

    public Ex7_23(Frame frm)
    {
        graphics = frm.getGraphics();
        width = frm.getSize().width;
        height = frm.getSize().height;
    }
public void drawJulia(){
    //比例后的坐标范围      (-1.5, 1.5)-(1.5, -1.5)

    final double a = 0.5;           //c=a+bi 为Julia集的参数
    final double b = 0.55;

    for( double x0 = -1.5; x0 < 1.5; x0+=0.01 )
    for( double y0 = -1.5; y0 < 1.5; y0+=0.01 ){
        double x=x0, y=y0;
        int n;
        for( n = 1; n<100; n++ ){
            double x2 = x * x - y * y + a;
            double y2 = 2 * x * y + b;
            x = x2;
            y = y2;
            if( x * x + y * y > 4 ) break;
```

```
                    }
                    pSet(x0, y0, n ); // 按n值来将 (x,y) 点进行着色
                }
        }
        public void pSet(double x, double y, int n){
                graphics.setColor( new Color( n* 0xff8855 ) );
                graphics.drawLine(
                        (int)(x*width/3 + width/2),
                        (int)(y*height/3 + height/2 ),
                        (int)(x*width/3 + width/2),
                        (int)(y*height/3 + height/2 )
                        );
        }
}
```

24. 求一配尔不定方程 $x^2 - Dy^2 = 1$ 的最小正整数解：

其中D为某个给定的常数。令 D=92，求其解。再令 D=29，求其解。这里都假定已知其解都在10000以内。

答：见程序。

```
public class Ex7_24
{
        public static void main(String[] args)
        {
                Solution( 29 );
                Solution( 92 );
        }

        public static void Solution( int D )
        {
                double x, y;
                for( y=1; ; y++ ){
                        x = Math.sqrt( 1 + D*y*y );
                        if( (int)x == x ){
                                System.out.println(
                                        (int)x + " ^2 - " + D +"*"+ (int)y + " ^2 = 1");
                                break;
                        }
                }
        }
}
```

25. 从键盘上输入 10个整数，并放入一个一维数组中，然后将其前 5个元素与后 5个元素对换，即：第 1个元素与第 10个元素互换， 第2个元素与第 9个元素互换……第 5个元素与第 6个元素互换。分别输出数组原来各元素的值和对换后各元素的值。

答：见程序。

```
import java.io.*;

class Ex7_25
{
        public static void main(String[] args)
        {
```

```
int [] a = new int[ 10 ];
for( int i=0; i<a.length; i++ )
{
        System.out.print( " 请输入第  " + (i+1)      + "个数 : " );
        a[i] = readOneIntger();
}

System.out.println( "  交换前为  :" );
print(a);

for( int i=0; i<a.length/2; i++ )
{
        int t;
        t = a[ a.length - i     -1];
        a[ a.length - i     - 1] = a[i];
        a[i] = t;
    }

        System.out.println( "  交换后为  :" );
        print(a);

}

static void print( int [] a)
{
        for( int i=0; i<a.length; i++ )
                System.out.print( a[i] + " " );
        System.out.println();
}

static int readOneIntger( )
{
        int n = 0;
        try{
                BufferedReader in = new BufferedReader(
                        new InputStreamReader( System.in ) );
                String s = in.readLine();
                        n = Integer.parseInt( s );
                }
                catch(IOException e){}
                return n;
        }


}
```

26．有一个 n*m 的矩阵，编写程序，找出其中最大的那个元素所在的行和列，并输出其值及行号和列号。

```
class    Ex7_26
{
        public static void main(String[] args)
        {
                int n = 5;
```

```
int m = 6;
int [][] a = new int[n][m];
for( int i=0; i<n; i++ )
        for( int j=0; j<m; j++ )
                a[i][j] = rnd(100);

for( int i=0; i<n; i++ )
{
        for( int j=0; j<m; j++ )
                System.out.print("\t" + a[i][j] );
        System.out.println("\n");
}


int max=a[0][0];
int p = 0;
        int q = 0;
        for( int i=0; i<n; i++ )
        {
                for( int j=0; j<m; j++ )
                {
                        if( max < a[i][j] )
                        {
                                max = a[i][j];
                                p=i;
                                q=j;
                        }
                }
        }

        System.out.println(" 第 "+(p+1)+" 行 "+(q+1)+" 列的值最大：  "+max);
}

public static int rnd( int max )
{
        return (int)(Math.random() * max ) + 1;
}
```

}

第 8章 Java 的多线程

1. 程序中怎样创建线程？

答：创建线程对象有两种方法，一是通过继承 Thread类，一是向 Thread类传递一个 Runnable 对象。

2. 程序中怎样控制线程？

答：启动一个线程：调用 start()方法。

终止或暂停一个线程：一般采取给线程设定一个标记变量的方法，来决定线程是否应该终止或暂停。

设定线程的优先级：使用　Thread对象的 setPriority(int priority)　方法。

3. 多线程之间怎样进行同步？

答：可以通过使用关键字　synchronized，使用对象的　wait()、notify() 方法来实现线程间的同步。

4. 编写一个程序，　用一个线程显示时间，　一个线程用来计算　（如判断一个大数是否是质数）　，当质数计算完毕后，停止时间的显示。

答：见程序。　import java.math.BigInteger;

```
import java.util.Random;

public class Ex8_4 {
	public static void main(String args[]) {
		new Ex8_4().test();
	}

public void test() {
	MyTask mytask = new MyTask();
	Thread thread = new Thread(mytask);
	thread.start();

	new Thread( new Clock() ).start();
}

boolean finished = false;

class MyTask implements Runnable {
	public void run() {
		for( int i=0; i<10; i++ ){
			doOnce();
			Thread.yield();
		}
		finished = true;
	}
	public void doOnce()
	{
		Random rnd = new Random();
		BigInteger prime =BigInteger.probablePrime(800,rnd);
		System.out.println(" 随机数是：  "+prime.toString());
		boolean bo=prime.isProbablePrime( 10000 ); //  验证
		System.out.println(bo?" 很可能是质数  ":" 不是质数  ");
	}
}

class Clock implements Runnable {
	public void run() {
		System.out.println("Start");
		while( ! finished ){
			System.out.print( "\r" + new java.util.Date());
			try{ Thread.sleep(1000); } catch( InterruptedException e ){}
			Thread.yield();
		}
```

```
        }
    }

    }
```

第 9章　流、文件及基于文本的应用

1. 字节流与字符流有什么差别？

答：按处理数据的类型，流可以分为字节流与字符流，它们处理的信息的基本单位分别是字节（ byte）与字符（ char）。

2. 节点流与处理流有什么差别？

答：节点流 (Node Stream): 直接与特定的地方（如磁盘、内存、设备等）相连，可以从 /向一个特定的地 (节点)读写数据。如文件流 FileReader。

处理流 (Processing Stream): 是对一个已存在的流的连接和封装，通过所封装的流的功能调用实现数据读 /写功能。处理流又称为过滤流。如缓冲处理流 BufferedReader。

3. 输入流与输出流各有什么方法？

答：输入流 InputStream 类最重要的方法是读数据的 read()方法。 read()方法功能是逐字节地二进制的原始方式读入数据。另外，它还有 skip(long n) 、reset()、available() 、close()方法等。

输出流 OutputStream 类的重要方法是 write() ，它的功能是将字节写入流中。另外，它还有 flush() 及 close() 方法。

4. 怎样进行文件及目录的管理？

答： Java支持文件管理和目录管理，它们都是由专门的 java.io.File 类来实现。每个 File 类的对象表示一个磁盘文件或目录，其对象属性中包含了文件或目录的相关信息，如名称、长度、所含文件个数等，调用它的方法则可以完成对文件或目录的常用管理操作，如创建、删除等。

5. 编写一个程序，从命令上行接收两个实数，计算其乘积。

答：参见程序。

```
import java.io.*;
public class Ex9_5
{
    public static void main(String[] args)
    {
        String s = "";
        double c = 0;
        double d = 0;
        try{
            BufferedReader in = new BufferedReader(
                new InputStreamReader( System.in ) );
            System.out.print(" 请输入一个数 : ");
            s = in.readLine();
            c = Double.parseDouble( s );
            System.out.print(" 请输入另一个数 : ");
            s = in.readLine();
            d = Double.parseDouble( s );
            System.out.println(" 这两个数的积为 : " + (c * d) );
        }catch(IOException e){}
    }
}
```

6. 编写一个程序，从命令行上接收两个文件名，比较两个文件的长度及内容。
答：参见程序。

```
import java.io.*;
public class Ex9_6
{
	public static void main(String[] args)
	{
		try{
			BufferedReader in = new BufferedReader(
				new InputStreamReader( System.in ) );
			System.out.print(" 请输入一个文件名    : ");
			String name1 = in.readLine();
			System.out.print(" 请输入另一个文件名    : ");
			String name2 = in.readLine();

			File file1 = new File( name1 );
			File file2 = new File( name2 );

			long len1 = file1.length();
			long len2 = file2.length();

			if( len1 != len2 ){
				System.out.println( " 这两个文件的长度不一样     ");
				return;
			}

			String text1 = readFileToEnd( file1 );
			String text2 = readFileToEnd( file2 );

			System.out.println( text1 );
			System.out.println( text2 );

			if( text1.equals(text2) ){
				System.out.println( " 这两个文件的内容相同     ");
			}else{
				System.out.println( " 这两个文件的内容不一样     ");
			}

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

	public static String readFileToEnd( File file ){
		StringBuffer text = new StringBuffer();
		try {
			BufferedReader in = new BufferedReader(
				new FileReader(file));
			String s = in.readLine();
			while ( s != null ) {
				text.append( s + "\n" );
```

```
                        s = in.readLine();
                    }
                    in.close();
                } catch (IOException e2) {
                    e2.printStackTrace();
                }
                return text.toString();
        }

}
```

7. 编写一个程序，能将一个 Java源程序中的空行及注释去掉。
答：参见程序。 import java.io.*;

```
public class Ex9_7 {
    public static void main (String[] args) {
        String infname = "CopyFileAddLineNumber.java";
        String outfname = "CopyFileAddLineNumber.txt";
        if( args.length >= 1 ) infname = args[0];
        if( args.length >= 2 ) outfname = args[1];

        try {
            File fin = new File(infname);
            File fout = new File(outfname);

            BufferedReader in = new BufferedReader(new FileReader(fin));
            PrintWriter out  = new PrintWriter(new FileWriter(fout));

            int cnt = 0;    //  行号
            String s = in.readLine();
            while ( s != null ) {
                cnt ++;
                s = deleteComments(s);          //去掉以 //开始的注释
                if( s.length() != 0 ){
                    out.println( s );           //写出非空行
                }
                s = in.readLine();              //读入
            }
            in.close();                         //  关闭缓冲读入流及文件读入流的连接 .
            out.close();
        } catch (FileNotFoundException e1) {
            System.err.println("File not found!" );
        } catch (IOException e2) {
            e2.printStackTrace();
        }
    }

    static String deleteComments( String s ) // 去掉以 //开始的注释
    {
        if( s==null ) return s;
        int pos = s.indexOf( "//" );
        if( pos<0 ) return s;
```

```
            return s.substring( 0, pos );
        }
    }
```

第10章　图形用户界面

1. 试列举出图形用户界面中你使用过的组件。

答：常用的容器组件如：边框窗体（　　Frame）、对话框（ Dialog ）、面板（ Panel）、及滚动面板（ScrollPane）等。常用的非容器组件，　如按钮 （Button ），标签（Label），文本类组件 （TextField 及TextArea ），复选按钮 (Checkbox) , 单选按钮组 (CheckboxGroup) , Choice(下拉列表), List(列表), 滚动条 (Scrollbar), 画布 (Canvas)等。

2. Java中常用的布局管理各有什么特点？

答：常用的布局管理有　　5种，即 FlowLayout ， BorderLayout ， CardLayout ， GridLayout 和 GridBagLayout ， 它们的特点如下：

FlowLayout 是容器 Pane和它的子类 Applet 缺省使用的布局管理器，它将将其中的组件按照加入　的先后顺序从左向右排列，一行排满之后就下转到下一行继续。

BorderLayout 是容器 Frame和Dialog 缺省使用的布局管理器，它把容器内的空间简单地划分为东、西、南、北、中五个区域。

CardLayout 的容器表面上可以容纳多个组件，但是实际上同一时刻容器只能从这些组件中选出一个来显示，就像一叠　–扑克牌 ‖每次只能显示最上面的一张一样，这个被显示的组件将占据所有的容器空间。

GridLayout 是使用较多的布局管理器，其基本布局策略是把容器的空间划分成若干　　行乘若干列的网格区域，组件就位于这些划分出来的小格中。

在 GridBagLayout 中，可以为每个组件指定其包含的网格个数，可以保留组件原来的大小，可以以任意顺序随意加入容器的任意位置，从而可以真正自由地安排容器中每个组件的大小和位置。

3. 简述 Java的事件处理机制。

答：在 Java中通过实现事件监听器（　Eventistener）来实现对事件的处理。事件监听器是一些事件的接口，这些接口是　　java.awt.AWTEventListener 的子类。接口中含有相关的方法，每个方法中可以编程来处理相关的事件。

每个界面对象在需要处理某种事件时，先进行事件的注册。注册的过程就是将界面对象与事件监听器联系在一起的过程。

4. 什么是事件源　?什么是监听者　?

答：事件源是指发出事件的界面对象。监听者是指处理这些事件的对象，它实现了一些相关事件处理方法的接口，在一定意义上监听者是具体处理这些事件的程序。

5. 列举 java． awt． event包中定义的事件类，并写出它们的继承关系。

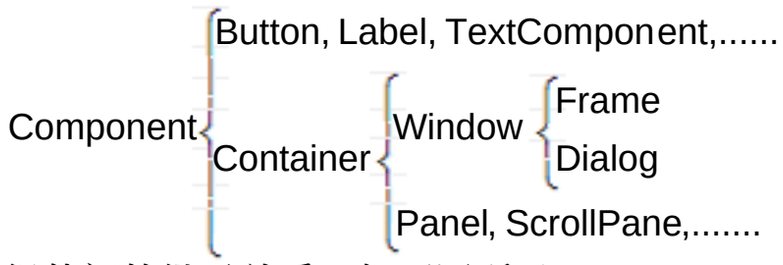
答：在 java.awt.event 包中有一系列的事件类，如　　KeyEvent 及 MouseEvent 等。这些事件类都是从 AWTEvent 类派生而来的。事件类之间的继承关系如下图所示。

6. 列举 GUI 的各种标准组件和它们之间的层次继承关系。

答：组件（ Component）分为容器（ Container）类和非容器类组件两大类。容器本身也是组件，但容器中可以包含其他组件，也可以包含其他容器。非容器类组件的种类较多，如按钮（Button ），标签（ Label） ,文本类组件 TextComponent 等等。

容器又分为两种顶层容器和非顶层容器两大类。顶层容器是可以独立的窗口，顶层容器的类是 Window ， Window 的重要子类是 Frame和 Dialog 。非顶层容器，不是独立的窗口，它们必须位于窗口之内，非顶层容器包括 Panel及 ScrollPane等， Panel的重要子类是 Applet 。其中， Panel 和 Applet 的容器都是无边框的； ScrollPanel 一组是可以自动处理滚动操作的容器； Window ，Frame，Dialog 和 FileDialog 是一组大都含有边框，并可以移动、放大、缩小、关闭的功能较强的容器。

AWT 组件的分类如下图所示：



组件间的继承关系，如下图所示：

7. Component类有何特殊之处 ?其中定义了哪些常用方法 ？

答： Component类是所有组件和容器的抽象父类，其中定义了一些每个容器和组件都可能用到的方法，较常用的有：

(1)public void add(PopupMenu popup)：在组件上加入一个弹出菜单，当用户用鼠标右键单击组件时将弹出这个菜单。

(2)public Color getBackground() ：获得组件的背景色。

(3)public Font getFont() ：获得组件使用的字体。

(4)public Color getForeground() ：获得组件的前景色。

(5)public Graphics getGraphics() ：获得在组件上绘图时需要使用的 Graphics对象。

(6)public void repaint(int x ， int y， int width ， int height) ：以指定的坐标点 (x， y)为左上角，重画组件中指定宽度 (width) 、指定高度 (height)的区域。

(7)public void setBackground(Color c) ：设置组件的背景色。

(8)public voidsetEnabled(boolean b) ：设置组件的使能状态。参数 b为真则组件使能，否则组件不使能。只有使能状态的组件才能接受用户输入并引发事件。

(9)public void setFont(Font f) ：设置组件使用的字体。

(10)public void setSize(int width ， int height) ：设置组件的大小。

(11)public void setVisible(boolean b) ：设置组件是否可见的属性。参数 b为真时， 组件在包括它的容器可见时也可见；否则组件不可见。

(12)public void setForeground(Color c) ：设置组件的前景色。

(13)public void requestFocus() ：使组件获得注意的焦点。

8. 将各种常用组件的创建语句、常用方法、可能引发的事件、需要注册的监听者和监听者需要重载的方法综合在一张表格中画出。

答：常用的组件与事件的关系如下表所示：

组 件	**Act**	**Adj**	**Cmp**	**Cnt**	**Foc**	**Itm**	**Key**	**Mou**	**MM**	**Txt**	**Win**
Button	Y		Y		Y		Y	Y	Y		
Canvas			Y		Y		Y	Y	Y		
Checkbox			Y		Y	Y	Y	Y	Y		
CheckboxMenuItem						Y					
Choice			Y		Y	Y	Y	Y	Y		
Component			Y		Y		Y	Y	Y		
Comtainer			Y	Y	Y		Y	Y	Y		
Dialog			Y	Y	Y		Y	Y	Y		Y
Frame			Y	Y	Y		Y	Y	Y		Y

Label			Y		Y		Y	Y	Y		
List	Y		Y		Y	Y	Y	Y	Y		
MenuItem	Y										
Panel			Y	Y	Y		Y	Y	Y		
Scrollbar		Y	Y		Y		Y	Y	Y		
ScrollPane			Y	Y	Y		Y	Y	Y		
TextArea			Y		Y		Y	Y	Y	Y	
TextField	Y		Y		Y		Y	Y	Y	Y	
Window			Y	Y	Y		Y	Y	Y		Y

注：

Act： Action 行动事件
Adj ： Adjustment 调整
Cmp： Component组件事件
Cnt： Container容器事件
Foc： Focus 焦点事件
Itm ： Item 条目事件
Key： Key键盘事件
Mou ： Mouse 鼠标事件
MM ： Mouse Motion 鼠标移动事件
Txt ： Text 文本事件
Win ： Window 窗口事件

9. 编写 Applet 包括一个标签、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。

答：见程序。

10. 编写 Applet 程序，画出一条螺旋线。

答：见程序。

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

public class Ex10_10 extends Applet
{
    public void paint(Graphics g){
        double w = getSize().width/2;
        double h = getSize().height/2;
        g.setColor( Color.blue );
        for( double th =0; th<h; th+=0.003){
            double r = th;
            double x = r * Math.cos( th ) + w;
            double y = r * Math.sin( th ) + h;
            g.drawLine( (int)x, (int)y, (int)x, (int)y);
        }
    }
}
```

11. 编写 Applet 程序，用 paint()方法显示一行字符串，Applet 包含两个按钮 “放大”和“缩小”，当用户单击“放大”时显示的字符串字体放大一号，单击“缩小”时显示的字符串字体缩小一号。

答：见程序。

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

```
public class    Ex10_11 extends Applet
{

        Button btn1 = new Button("  放大 ");
        Button btn2 = new Button("  缩小 ");

        public void init()
        {
                setLayout( new FlowLayout() );
                add( btn1 );
                add( btn2 );
                btn1.addActionListener( new BtnActionAdapter() );
                btn2.addActionListener( new BtnActionAdapter() );
        }

        private int fontSize = 12;

        public void paint(Graphics g){
                double w = getSize().width/2;
                double h = getSize().height/2;
                g.setColor( Color.blue );
                g.setFont( new Font( " 宋体 ", Font.PLAIN, fontSize ) );
                g.drawString( " 文字的大小  " + fontSize , 10, 15 );
        }

        class BtnActionAdapter implements ActionListener
        {
                public void actionPerformed( ActionEvent e )
                {
                        if( e.getSource() == btn1 ) fontSize += 2;
                        if( e.getSource() == btn2 ) fontSize -= 2;
                        if( fontSize <= 5 ) fontSize = 5;
                        Ex10_11.this.repaint();
                }
        }

}
```

12．编写 Applet 程序，包含三个标签，其背景分别为红、黄、蓝三色。

答：见程序。

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

public class    Ex10_12 extends Applet
{
        Label lb1 = new Label("  红 ");
        Label lb2 = new Label("  黄 ");
        Label lb3 = new Label("  蓝 ");

        public void init()
        {
```

```
            setLayout( new FlowLayout() );
            add( lb1 );
            add( lb2 );
            add( lb3 );
            lb1.setBackground( Color.red );
            lb2.setBackground( Color.yellow );
            lb3.setBackground( Color.blue );
        }
    }
```

13. 使用 Checkbox 标志按钮的背景色， 使用 CheckboxGroup 标志三种字体风格， 使用 Choice 选择字号，使用 List 选择字体名称，由用户确定按钮的背景色和前景字符的显示效果。

答：见程序。

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

public class  Ex10_13 extends Applet implements ActionListener
{
    Checkbox chkBack = new Checkbox("  背景为黄色  ");

    CheckboxGroup cbgStyle = new CheckboxGroup();

    Choice chcSize = new Choice();

    List lstName = new List(4, false);
    Button btnOk = new Button("  确定 ");

    public void init()
    {
        add( chkBack );

        add(new Checkbox(" 常规 ", cbgStyle, false));
        add(new Checkbox(" 粗体 ", cbgStyle, true));
        add(new Checkbox(" 斜体 ", cbgStyle, false));

        chcSize.add( "9" );
        chcSize.add( "11" );
        chcSize.add( "15" );
        add( chcSize );

        lstName.add( " 宋体 ");
        lstName.add( " 楷体 ");
        lstName.add( " 黑体 ");
        add( lstName );

        add( btnOk);

        btnOk.addActionListener( this );
    }

    public void actionPerformed(ActionEvent e)
```

```
{
        boolean back = chkBack.getState();

        int fontStyle = Font.PLAIN;
        Checkbox chkStyle = cbgStyle.getSelectedCheckbox();
        if( chkStyle.getLabel() == "   粗体 " )
                fontStyle = Font.BOLD;
        if( chkStyle.getLabel() == "   斜体 " )
                fontStyle = Font.ITALIC;

        int fontSize = Integer.parseInt( chcSize.getSelectedItem() );

        String fontName = lstName.getSelectedItem();
        if( fontName == null ) fontName = "    宋体 ";

                Font font = new Font( fontName, fontStyle, fontSize );

                if( back ) btnOk.setBackground( Color.yellow );

                btnOk.setFont( font );

        }
}
```

14．使用滚动条：编写一个　Applet 包含一个滚动条，在　Applet 中绘制一个圆，用滚动条滑块显示的数字表示该圆的直径，当用户拖动滑块时，圆的大小随之改变。

答：见程序。

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

public class    Ex10_14 extends Applet
{

        Scrollbar scrl = new Scrollbar(Scrollbar.HORIZONTAL);

        public void init()
{
        //scrl.setBounds( 10,10, getSize().width-20, 10 );
        scrl.setMaximum( getSize().width );
        scrl.setValue( rad * 2);
        scrl.addAdjustmentListener( new MyListener() );
        setLayout( new BorderLayout() );
        add( scrl, BorderLayout.NORTH );
}

private int rad = 12;

public void paint(Graphics g){
        int w = getSize().width/2;
        int h = getSize().height/2;
        g.setColor( Color.blue );
```

```
            g.drawOval( w-rad, h-rad, rad*2, rad*2 );
        }

        class MyListener implements AdjustmentListener
        {
            public void adjustmentValueChanged( AdjustmentEvent e )
            {
                rad = scrl.getValue()/2;
                if( rad <= 5 ) rad = 5;
                Ex10_14.this.repaint();
            }
        }

}
```

15. 编写一个 Applet 响应鼠标事件， 用户可以通过拖动鼠标在 Applet 中画出矩形， 并在状态条显示鼠标当前的位置。 使用一个 Vector对象保存用户所画过的每个矩形并显示、 响应键盘事件，当用户击键 q时清除屏幕上所有的矩形。

答：见程序。

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

public class    Ex10_15 extends Applet
{
    public void init()
    {
        this.addMouseListener( new MyMouseListener() );
        this.addKeyListener( new MyKeyListener() );
    }

    private Vector rects = new V ector();
    private Point down = null;

    public void paint(Graphics g){
        g.setColor( Color.blue );
        for( int i=rects.size()-1; i>=0; i-- ){
            Rectangle r = (Rectangle)rects.get(i);
            g.drawRect( (int)r.getX(), (int)r.getY(),
                (int)r.getWidth(), (int)r.getHeight() );
        }
        g.drawString( " 有"+rects.size()+" 个矩形 ", 10,15 );
        System.out.println("R");
}

class MyMouseListener extends MouseAdapter
{
    public void mousePressed(MouseEvent e)
    {
        down = e.getPoint();
    }
    public void mouseReleased(MouseEvent e)
```

```
{
        Point up = e.getPoint();
        if( down==null || up.equals(down) ){
                down = null;
                return;
        }else{
                rects.add( new Rectangle(
                        (int)down.getX(), (int)down.getY(),
                        (int)(up.getX()-down.getX()),
                        (int)(up.getY()-down.getY())
                ));
                down = null;
                Ex10_15.this.repaint();
        }
        System.out.println("M");
}
        }

        class MyKeyListener extends KeyAdapter
        {
                public void keyTyped(KeyEvent e)
                {
                        if( e.getKeyChar() == 'q' ){
                                rects = new Vector();
                                Ex10_15.this.repaint();
                        }
                        System.out.println("K");
                }
        }
}
```

16. 编写 Applet 程序实现一个计算器，包括十个数字 (0～9)按钮和四个运算符 (加、减、乘、除)按钮，以及等号和清空两个辅助按钮，还有一个显示输入输出的文本框。试用 BorderLayout 和GridLayout 实现。

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

public class  Ex10_16 extends Applet implements ActionListener
{
        double  d0,d1,d2,result;
        boolean  flag=true;
        String  s,oper;
        TextField  tf1;
        Panel  p=new Panel();

        public void init()
        {

                int i;
                result=0;
                s=new String();
                oper=new String("+");
                tf1=new TextField("",15);
```

```
Button[] b=new Button[21];
for(i=1;i<21;i++)
{
        b[i]=new Button();
        b[i].setFont(new Font("  仿宋 ",0,16));
}
b[1].setLabel(" 退格 ");
b[2].setLabel("CE");
b[3].setLabel("C");
b[4].setLabel("/");
b[5].setLabel("7");
b[6].setLabel("8");
b[7].setLabel("9");
b[8].setLabel("*");
b[9].setLabel("4");
b[10].setLabel("5");
b[11].setLabel("6");
b[12].setLabel("-");
b[13].setLabel("1");
b[14].setLabel("2");
b[15].setLabel("3");
b[16].setLabel("+");
b[17].setLabel("0");
b[18].setLabel("+/-");
b[19].setLabel(".");
b[20].setLabel("=");
p.setLayout(new GridLayout(5,4));
p.setBackground(new Color(80,30,100));
for(i=1;i<21;i++)
{
        p.add(b[i]);
        b[i].addActionListener(this);
        b[i].setBackground(new Color(20,130,180));
        b[i].setForeground(Color.yellow);
}
for(i=1;i<4;i++)
{
        b[i].setBackground(new Color(120,180,170));
        b[i].setForeground(Color.blue);
}
for(i=1;i<=4;i++)
{
        b[i*4].setBackground(new Color(120,180,170));
        b[i*4].setForeground(Color.red);
}
b[20].setBackground(Color.red);
setLayout(new BorderLayout());
add("North",tf1);
add("Center",p);
}


public void actionPerformed(ActionEvent e)
```

```
{
        String i1=tf1.getText();
        s=e.getActionCommand();
        if(s=="0"| s=="1"|s=="2"|s=="3"|s=="4"|s=="5"|s=="6"|s=="7"|s=="8"|s=="9"|s==".")
        {
            if(flag) tf1.setText(i1+s);
           else
           {
                tf1.setText(s);
                flag=true;
           }
        }
        else if(s=="+"|s=="-"|s=="*"|s=="/")
        {
            result=Double.parseDouble(i1);
            flag=false;
            oper=s;
        }
        else if(s=="=")
        {
            d0=Double.parseDouble(i1);
            if(oper=="+")      result+=d0;
            if(oper=="-")      result-=d0;
            if(oper=="*")      result*=d0;
            if(oper=="/")      result/=d0;
            tf1.setText(Double.toString(result));
            flag=false;
        }
        else if(s=="CE")
        {
            tf1.setText("");
            flag=false;
        }
        else if(s=="C")
        {
            result=0;
            tf1.setText("");
            flag=false;
        }
        else if(s==" 退格 ")
        {
            String ss=tf1.getText();
            int i=ss.length();
            ss=ss.substring(0,i-1);
            tf1.setText(ss);
        }
        else if(s=="+/-")
        {
            d2=-1*Double.parseDouble(tf1.getText());
            tf1.setText(Double.toString(d2));
        }
    }
}
```

答：见程序。

17. Panel与Applet 有何关系 ?Panel在 Java程序里通常起到什么作用 ?

答： Panel类是 Applet 类的父类。 Panel在Java程序里通常用作容器，用于将其他对象进行布局。

18. 为什么说 Frame 是非常重要的容器 ?为什么使用 Frame 的程序通常要实现 WindowListener? 关闭 Frame有哪些方法 ?

答：Frame是可以独立存在的顶层容器，相当于我们在 Windows 中看见的独立窗口，所以它是相当重要的。由于 Frame类自己没有自动关闭的功能，所以通常要实现 WindowListener 来处理关闭事件。

常用的关闭窗口的方法有三个：一个是设置一个按钮，当用户点击按钮时关闭窗口；第二个方法是对 WINDOWS_CLOSING 事件做出响应，关闭窗口；第三个方法是使用菜单命令。前一种方法需要专门的按钮，而后一种方法实现 WindowListener 接口所需的代码较多，无论使用何种方法，都需要用到关闭 Frame的dispose()方法。

19. 练习使用列表框及组合框。

答：略。（可参见第 13题）。

20. Swing 组件与 AWT 件有何区别。

答：Swing 是第二代 GUI 开发工具集， 与AWT 是第一代 GUI 开发工具集。 与AWT 相比， Swing 具有更好的可移植性， Swing 提供了更完整的组件，增加了许多功能。此外， Swing 引入了许多新的特性和能力， 如：所有的组件都很小巧的 （轻量级的），支持双缓存， 支持拖放， 支持文本、图形、工具提示 (Tooltip) ，新的布局管理，更多的组件等等。

21. 绘出以下函数的曲线：

y = 5 * sin(x) + cos(3*x)

y = sin(x) + sin(6*x)/10

答：见程序。

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

public class Ex10_21 extends Applet
{
public void paint(Graphics g){
        double w = getSize().width;
        double h = getSize().height;
        g.setColor( Color.red );
        for( double x =0; x<w/10; x+=0.003){
                double y = 5*Math.sin( x ) + Math.cos(3*x);
                double xx = x * 10;
                double yy = y * h/22 + h/4;
                g.drawLine( (int)xx, (int)yy, (int)xx, (int)yy);
        }
        g.setColor( Color.blue );
        for( double x =0; x<w/10; x+=0.003){
                double y = Math.sin( x ) + Math.cos(6*x)/10;
                double xx = x * 10 ;
                double yy = y * h/5 + h*3/4;
```

```
                g.drawLine( (int)xx, (int)yy, (int)xx, (int)yy);
            }
        }
}
```

22. 绘出以下函数的曲线：

r=cos(2 θ)

r=cos(3 θ)

答：见程序。

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

public class Ex10_22 extends Applet
{
    public void paint(Graphics g){
        double w = getSize().width/2;
        double h = getSize().height/2;
        g.setColor( Color.red );
        for( double th =0; th<10; th+=0.003){
            double r = Math.cos( 2 * th ) * h;
            double x = r * Math.cos( th ) + w/2;
            double y = r * Math.sin( th ) + h;
            g.drawLine( (int)x, (int)y, (int)x, (int)y);
        }
        g.setColor( Color.blue );
        for( double th =0; th<10; th+=0.003){
            double r = Math.cos( 3 * th ) * h;
            double x = r * Math.cos( th ) + w*3/2;
            double y = r * Math.sin( th ) + h;
            g.drawLine( (int)x, (int)y, (int)x, (int)y);
        }
    }
}
```

23. 根据本章的所学习的内容用 JavaApplication 编写一个模拟的文本编辑器。 给文本编辑器增加设字体字号的功能。

答：见程序。

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

class TextEditorFrame extends JFrame
{

    File file = null;
    Color color = Color.black;

    TextEditorFrame(){
        initTextPane();
        initMenu();
        initAboutDialog();
```

```
        initToolBar();
}

void initTextPane(){ //  将文本框放入有滚动对象，并加入到        Frame中
        getContentPane().add( new JScrollPane(text) );
}

JTextPane text = new JTextPane();    //文本框
JFileChooser filechooser = new JFileChooser(); //  文件选择对话框
JColorChooser colorchooser = new JColorChooser(); //  颜色选择对话框
JFontChooser fontchooser = new JFontChooser(); // 颜色选择对话框
JDialog about = new JDialog(this); //   关于对话框
JMenuBar menubar = new JMenuBar(); //  菜单

JMenu [] menus = new JMenu[] {
        new JMenu("File"),
        new JMenu("Edit"),
        new JMenu("Help")
};
JMenuItem menuitems [][] = new JMenuItem[][]{{
        new JMenuItem("New"),
        new JMenuItem("Open..."),
        new JMenuItem("Save..."),
        new JMenuItem("Exit")},{
        new JMenuItem("Copy"),
        new JMenuItem("Cut"),
        new JMenuItem("Paste"),
        new JMenuItem("Color..."),
        new JMenuItem("Font...")},{
        new JMenuItem("About")}
};

void initMenu(){       //初始化菜单
        for( int i=0; i<menus.length; i++ ){
                menubar.add( menus[i] );
                for( int j=0; j<menuitems[i].length; j++ ){
                        menus[i].add( menuitems[i][j] );
                        menuitems[i][j].addActionListener( action );
                }
        }
        this.setJMenuBar( menubar );
}

ActionListener action = new ActionListener(){ //     菜单事件处理
        public void actionPerformed( ActionEvent e ){
                JMenuItem mi = (JMenuItem)e.getSource();
                String id = mi.getText();
                if( id.equals("New" )){
                        text.setText("");
                        file = null;
                }else if( id.equals("Open...")){
                        if( file != null ) filechooser.setSelectedFile( file );
```

```
            int returnVal = filechooser.showOpenDialog(
                    TextEditorFrame.this);
            if(returnVal == JFileChooser.APPROVE_OPTION) {
                    file = filechooser.getSelectedFile();
                    openFile();
            }
    }else if( id.equals("Save...")){
            if( file != null ) filechooser.setSelectedFile( file );
            int returnVal = filechooser.showSaveDialog(
                    TextEditorFrame.this);
            if(returnVal == JFileChooser.APPROVE_OPTION) {
                    file = filechooser.getSelectedFile();
                    saveFile();
            }
    }else if( id.equals("Exit")){
            System.exit(0);
                    }else if( id.equals("Cut")){
                            text.cut();
                    }else if( id.equals("Copy")){
                            text.copy();
                    }else if( id.equals("Paste")){
                            text.paste();
                    }else if( id.equals("Color...")){
                            color = JColorChooser.showDialog(
                                    TextEditorFrame.this, "", color );
                            text.setForeground(color);
                    }else if( id.equals("Font...")){
                            fontchooser.setVisible( true );
                            Font font = fontchooser.getFont();
                            text.setFont(font);
                    }else if( id.equals("About")){
                            about.setSize(100,50);
                            about.setVisible(true);
                    }
            }
    };

    void saveFile(){ //  保存文件，将字符写入文件
            try{
                    FileWriter fw = new FileWriter( file );
                    fw.write( text.getText() );
                    fw.close();
            }catch(Exception e ){ e.printStackTrace(); }
    }
    void openFile(){ //  读入文件，并将字符置入文本框中
            try{
                    FileReader fr = new FileReader( file );
                    int len = (int) file.length();
                    char [] buffer = new char[len];
                    fr.read( buffer, 0, len );
                    fr.close();
                    text.setText( new String( buffer ) );
            }catch(Exception e ){ e.printStackTrace(); }
```