面向对象程序设计—Java 语言习题答案

第 1 章

一、选择题

1．下列选项中，不属于 Java 语言特点的一项是（ C ）。

（A）分布式　　（B）安全性

（C）编译执行　　（D）面向对象

【解析】Java 程序采用解释执行的方法。在系统编译运行 Java 程序时，Java 编译器将 Java 程序转化为字节码，在运行时，解释器将编译得到的字节码进行解释执行。

2．在 Java 语言中，（ C ）是最基本的元素？

（A）方法　　（B）包

（C）对象　　（D）接口

【解析】构成 Java 程序的基本元素类（抽象的对象）。

3．编译一个定义了 3 个类和 10 个方法的 Java 源文件后，会产生（ D ）个字节码文件？扩展名是（ D ）？

（A）13 个字节码文件，扩展名为.class　　（B）1 个字节码文件，扩展名为.class

（C）3 个字节码文件，扩展名为.java　　（D）3 个字节码文件，扩展名为.class

【解析】源文件中的每一个类编译后都会生成一个字节码文件，字节码文件的扩展名是.class。

4．在创建 Applet 应用程序时，需要用户考虑问题是（ B ）。

（A）窗口如何创建　　（B）绘制的图形在窗口中的位置

（C）程序的框架　　（D）事件处理

【解析】创建 Applet 程序时必须继承系统类 Applet，而 Applet 类中已经包含了如何创建窗口以及事件处理等内容，这类程序的框架也都是固定的，而绘制图形在窗口中的位置则需要由用户确定。

5．Java 语言属于（ B ）种语言？

（A）面向机器的语言　　（B）面向对象的语言

（C）面向过程的语言　　（D）面向操作系统的语言

【解析】Java 语言是一种纯面向对象的语言。

6．下列关于 Application 和 Applet 程序的说法中不正确的一项是（ B ）。

（A）Application 使用解释器 java.exe　　（B）Application 不使用独立的解释器

（C）Applet 在浏览器中运行　　（D）Applet 必须继承 Java 的 Applet 类

【解析】Application 程序包含 main()方法，它是一种独立执行的程序，因此必须使用独立的解释器解释执行。

7．下列选项中，不属于 Java 核心包的一项是（ A ）。

（A）javax.swing　　（B）java.io

（C）java.utile　　（D）java.lang

【解析】凡是以 java 开头的包都是 Java 核心包，以 javax 开头的包则属于 Java 扩展包。

8．下列描述中，不正确的是（ A ）。

（A）不支持多线程　　（B）一个 Java 源文件不允许有多个公共类

（C）Java 通过接口支持多重继承　（D）Java 程序分为 Application 和 Applet 两类

【解析】Java 是支持多线程的语言。

9．阅读下列代码，选出该代码段正确的文件名（ C ）。

```
class A{
    void method1(){
        System.out.println("Method1 in class A");
    }
}
public class B{
    void method2(){
        System.out.println("Method2 in class B");
    }
    public static void main(String[] args){
        System.out.println("main() in class B");
    }
}
```

（A）A.java　（B）A.class

（C）B.java　（D）B.class

【解析】Java 源文件名必须和公共类的名字完全一样，源文件的扩展名为.java。

10．编译下面源程序会得到哪些文件（ D ）？

```
class A1{
}
class A2{
}
public class B{
    public static void main(String[] args){
    }
}
```

（A）只有 B.class 文件　（B）只有 A1.class 和 A2.class 文件

（C）编译不成功　（D）A1.class、A2.class 和 B.class 文件

【解析】由于该程序包含 3 个类，每个类编译后都会生成 1 个字节码文件，因此编译后会生成以这 3 个类名命名的字节码文件。

二、填空题

1．Java 程序的编译和执行模式包括 2 点，是__半编译__和__半解释__。

2．Java 语言支持__TCP/IP__协议，从而使得 Java 程序在分布式环境中能够很方便地访问处于不同地点的__对象__。

3．开发 Java 程序的一般步骤是：源程序编辑、__生成字节码__和__解释执行__。

4．每个 Java Application 程序可以包括许多方法，但是必须有且只能有一个__main()__方法，统一格式为__public static void main(String[] args)__，它是程序执行的入口。

5.JVM 把字节码程序与各种不同的__操作系统__和__硬件__分开，使得 Java 程序独立于平台。

6．在 Java 程序中，能在 WWW 浏览器上运行的是__Applet__程序。

7．Java 源程序文件和字节码文件的扩展名分别为__.java__和__.class__。

8．如果在 Java 程序中需要使用 java.utile 包中的所有类，则应该在程序开始处加上__import java.utitle.*__语句。

三、编程题

1．编写一个 Java Application 类型的程序，输出“This is my first Java Application!”。

【编程分析】 要编写 Java Application 类型的程序，需要在 JCreator 中创建一个 Java Application 类型的工程，这时 JCreator 会自动创建程序框架，该框架包含一个公共类，其中包含一个 main()方法。删除 main()方法中自动生成的代码，编写自己的代码即可。由于该程序要求输出一句话，因此在 main()方法中直接调用 System.out.println()或 System.out.print()方法即可实现。

【参考程序】

```
public class X3_1 {
    public static void main(String args[]){
        System.out.println("This is my first Java Application!");
    }
}
```

【运行结果】

This is my first Java Application!

2．编写一个 Java Applet 类型的程序，输出“这是我的第一个 Java Applet 类型的程序”。

【编程分析】 要编写一个 Java Applet 类型的程序，首先利用 JCreator 创建一个 Java Applet 类型的工程，该工程自动创建两个文件，一个源文件（.java 文件）和一个 HTML 文件（.html 文件），源文件中包含一个公共类，其中包含两个方法，即 init()方法和 paint()方法，由于该程序只是要求输出一句话，因此只要将 paint()方法中 g.drawString()方法的第一个参数修改为要输出的内容即可。如果不是利用 JCreator 开发工具，而是利用 J2SDK，则用户需要自己编写 HTML 文件。

【参考程序】

```
import java.awt.*;
import java.applet.*;
public class X3_2 extends Applet {
    public void paint(Graphics g) {
        g.drawString("这是我的第一个 Java Applet 类型的程序",50,50);
    }
}
```

【运行结果】

第 2 章

一、选择题

1．下列（ D ）是合法的标识符？

（A）12class　　（B）void　　（C）-5　　（D）_blank

【解析】根据 Java 标识符的构成规则确定。

2．下列（ B ）不是 Java 中的保留字？

（A）if　　（B）sizeof　　（C）private　　（D）null

【解析】根据 Java 保留字表进行选择。

3．下列（ C ）不是合法的标识符？

（A）$million　　（B）$_million　　（C）2$_million　　（D）$2_million

【解析】根据 Java 标识符的构成规则确定。

4．下列选项中，（ B ）不属于 Java 语言的基本数据类型？

（A）整数型　　（B）数组　　（C）浮点型　　（D）字符型

【解析】根据 Java 语言的基本数据类型包含的类别选取。

5．下列关于基本数据类型的说法中，不正确的一项是（ D ）。

（A）boolean 类型变量的值只能取真或假

（B）float 是带符号的 32 位浮点数

（C）double 是带符号的 64 位浮点数

（D）char 是 8 位 Unicode 字符

【解析】Java 中的字符采用的是 16 位的 Unicode 字符。

6．下列关于基本数据类型的取值范围的描述中，正确的一个是（ B ）。

（A）byte 类型的取值范围是-128～128　（B）boolean 类型的取值范围是真或假

（C）char 类型的取值范围是 0～65536　（D）short 类型的取值范围是-32767～32767

【解析】根据每种类型占几个字节即可推算出其取值范围，如 byte 类型占 1 个字节，即共包含 2^8 个数值，其取值范围范围应为-128～127，其他类型也是如此。

7．下列关于 Java 语言简单数据类型的说法中，正确的一项是（ A ）。

（A）以 0 开头的整数代表 8 进制整型常量

（B）以 0x 或 0X 开头的整数代表 8 进制整型常量

（C）boolean 类型的数据作为类成员变量的时候，相同默认的初始值为 true

（D）double 类型的数据占计算机存储的 32 位

【解析】根据每种数据类型的特性进行判断。

8．下列 Java 语句中，不正确的一项是（ C ）。

（A）$e, a, b = 10;　　（B）char c, d = ‘a’;

（C）float e = 0.0d;　　（D）double c = 0.0f;

【解析】不能将 double 类型的常量赋值给 float 类型的变量。

9．在编写 Java 程序时，如果不为类的成员变量定义初始值，Java 会给出它们的默认值，下列说法中不正确的一个是（ D ）。

（A）byte 的默认值是 0　　（B）boolean 的默认值是 false

（C）char 类型的默认值是’\0’　　（D）long 类型的默认值是 0.0L

【解析】long 类型的默认值是 0L，而不是 0.0L。

10．下列语句中不正确的一个是（ B ）。

（A）float f = 1.1f;　　（B）byte b = 128;

（C）double d = 1.1/0.0;　　（D）char c = (char)1.1f;

【解析】bytel 类型变量的取值范围是-128～127。

11．下列表达式 1+2+ "aa"+3 的值是（ B ）。

（A）"12aa3"　　（B）"3aa3 "　　（C）"12aa"　　（D）"aa3"

【解析】整数和整数相加得到两个整数的和，而整数和字符串相加得到的是字符串。

12．已知 y=2, z=3, n=4，则经过 n=n+ -y*z/n 运算后 n 的值为（ A ）。

（A）3　　（B）-1　　（C）-12　　（D）-3

【解析】根据运算符的优先级，该表达式相当于 n=n+ ((-y)*z)/n。

13．已知 a=2, b=3，则表达式 a%b*4%b 的值为（ A ）。

（A）2　　（B）1　　（C）-1　　（D）-2

【解析】根据运算符的优先级，该表达式相当于((a%b)*4)%b。

14．已知 x=2, y=3, z=4，则经过 z- = --y – x--运算后，z 的值为（ D ）。

（A）1　　（B）2　　（C）3　　（D）4

【解析】在表达式运算过程中，--y 的值变为 2，x—的值还是 2，等号右侧运算后的值为 0，因此 z 的值没有变化。

15．表达式(12==0) && (1/0 < 1)的值为（ B ）。

（A）true　　（B）false　　（C）0　　（D）运行时抛出异常

【解析】由于(12==0)的值为 false，因此整个表达式发生短路运算，即(1/0 < 1)就没有参与运算，整个表达式的值为 false。

16．设有类型定义 short i=32; long j=64; 下面赋值语句中不正确的一个是（ B ）

（A）j=i;　　（B）i=j;　　（C）i=(short)j;　　（D）j=(long)i;

【解析】long 类型的数据不能自动转变为 short 类型，因此不能将 long 类型的变量直接赋值给 short 类型。

17．现有 1 个 char 类型的变量 c1=66 和 1 个整型变量 i=2，当执行 c1=c1+(char)i;语句后，c1 的值为（ D ）。

（A）'d'　　（B）'D'　　（C）68　　（D）语句在编译时出错

【解析】两个字符型的数据相加，得到的是一个整数，而如果把整数再赋值给一个字符型变量则会在编译时出错。

18．下列说法中，正确的一项是（ D ）。

（A）字符串"\\abcd"　的长度为 6　　（B）False 是 Java 的保留字

（C）123.45L 代表单精度浮点型　　（D）False 是合法的 Java 标识符

【解析】Java 语言对字符的大小写是敏感的，False 不是 false，因此 False 是合法的 Java 标识符。

19．以下的变量定义语句中，合法的是（ D ）

（A）float _*5 = 123.456F;　　（B）byte $_b1 = 12345;

（C）int _long_ = 123456L;　　（D）double d = Double.MAX_VALUE;

【解析】（A）中_*5 不是合法的标识符，（B）中 12345 超出 byte 范围，（C）中不能将 long 类型的常量赋值给 int 型的变量。

20．下列关于运算符优先级的说法中，不正确的一个是（ C ）

（A）运算符按照优先级顺序表进行运算

（B）同一优先级的运算符在表达式中都是按照从左到右的顺序进行运算的

（C）同一优先级的运算符在表达式中都是按照从右到左的顺序进行运算的

（D）括号可以改变运算的优先次序

【解析】同一优先级的运算符在表达式中都是按照从左到右的顺序进行运算的。

二、填空题

1. 变量是 Java 程序的基本存储单元之一，变量的主要类型包括 2 大类：字符型 和 数值型 。
2．Java 语言的整数类型变量和常量一样，各自都包括 4 种类型的数据，它们分别是 byte、int 、 short 和 long。
3． boolean 类型数据不可以做类型转换。
4．在 Java 语言的基本数据类型中，占存储空间最少的类型是 boolean ，该类型占用的存储空间为 1 位。
5．Java 语言中的 保留字 具有特殊意义和作用，不能作为普通标识符使用。
6．在 Java 语言中，浮点类型数据属于实型数据，可以分为 单精度 和 双精度 两种。
7．char 类型的数据可以表示的字符数共为 65536 。
8．定义初始值为 10 的 8 次方的常整型变量 iLong 的语句是 final iLong = 100000000L 。
9．Java 语言中的数据类型转换包括 自动转换 和 强制转换 两种。
10．Java 中的字符采用的是 16 位的 Unicode 编码。
11．数据类型中存储空间均为 64 位的两种数据类型是 long 和 double 。
12．表达式 9*4/ -5%5 的值为 -2 。（十进制表示）
13．表达式 5&2 的值为 0 。（十进制表示）
14．表达式 42<<4 的值为 672 。（十进制表示）
15．表达式 11010011>>>3 的值为 11010 。（二进制表示）
16．表达式 7|3 的值为 7 。（十进制表示）
17．表达式 10^2 的值为 8 。（十进制表示）
18．Java 语言中的逻辑与(&&)和逻辑或(||)运算采用 短路 方式进行运算。
19．若 a、b 为 int 型变量，并且已分别赋值为 5 和 10，则表达式(a++)+(++b)+a*b 的值为 82 。
20．假设 i=10, j=20, k=30，则表达式 !(i<j+k) || !(i+10<=j) 的值为 false 。

三、编程题

1．编写一个 Java Application 类型的程序，定义一个 byte 类型的变量 b，并从键盘上给它赋值为-100 和 100 时，输出该变量的值。
【编程分析】要实现键盘输入一个 byte 类型的变量，首先创建输入流对象，再利用该对象的 readLine()方法输入字符串，然后利用 Byte 类的 parseByte()方法将输入的字符串转化为字节类型，最后通过 System.out.println()方法输出该变量。
【参考程序】

```
import java.io.*;
public class X2_3_1 {
    public static void main(String[] args) throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        byte b ;
        String str = br.readLine();
        b = Byte.parseByte(str);
        System.out.println(b);
        b = Byte.parseByte(str);
       System.out.println(b);
```

```
        }
    }
```
【运行结果】

-200

```
Exception in thread "main" java.lang.NumberFormatException
        at java.lang.Byte.parseByte(Byte.java:124)
        at java.lang.Byte.parseByte(Byte.java:79)
        at EX11_1.main(EX11_1.java:8)
```

注意：给 b 赋值的范围只能在-128 至 127 之间，如果超出这个范围，则发生例外。

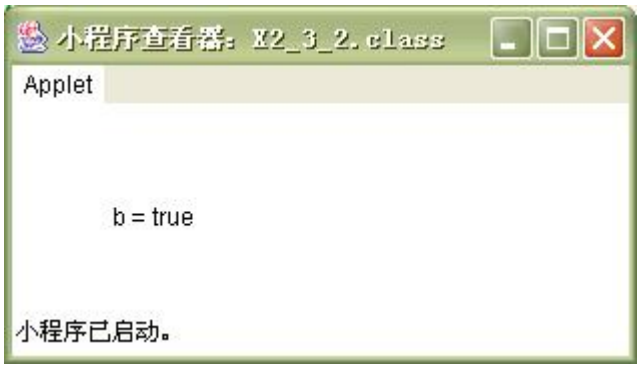
2．编写一个 Java Applet 类型的程序，计算输出表达式 12+5>3||12-5<7 的值。

【编程分析】由于表达式 12+5>3||12-5<7 的最终结果是 boolean 类型，因此可以将该表达式赋值给一个 boolean 类型的变量，然后输出该变量的值。

【参考程序】

```
import java.awt.*;
import java.applet.*;
public class X2_3_2 extends Applet {
    public void paint(Graphics g) {
        boolean b = 12+5>3||12-5<7;
        g.drawString("b = "+b, 50, 60 );
    }
}
```

【运行结果】



3．编写一个 Java Application 类型的程序，从键盘上输入三角形的三条边的长度，计算三角形的面积和周长并输出。根据三角形边长求面积公式如下：

$area = \sqrt{s*(s-a)*(s-b)*(s-c)}$，其中 a、b、c 为三角形的三条边，s=(a+b+c)/2。

【编程分析】该程序由于涉及到数据输入，因此首先建立输入流对象，输入三角形三条边 a、b、c 的值，然后求出中间变量 s 的值，最后利用数学方法 Math.sqrt()方法求出三角形的面积并输出。

【参考程序】

```
import java.io.*;
public class X2_3_3 {
    public static void main(String[] args) throws IOException{
```

```
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        double a, b, c, s, area;
        String str;
        str = br.readLine();
        a = Double.parseDouble(str);
        str = br.readLine();
        b = Double.parseDouble(str);
        str = br.readLine();
        c = Double.parseDouble(str);
        s = (a+b+c)/2.0;
        area = Math.sqrt(s*(s-a)*(s-b)*(s-c));
        System.out.println("area = "+area);
    }
}
```

【运行结果】

3

4

5

area = 6.0

注意：输入的三角形的三条边必须满足三角形的构成规则，如果不满足则输出错误结果。

4．编写一个 Java Application 类型的程序，从键盘上输入摄氏温度 C，计算华氏温度 F 的值并输出。其转换公式如下：

F = (9 / 5) * C + 32

【编程分析】该程序和上一个程序类似，在此不再赘述。

【参考程序】

```
import java.io.*;
public class X2_3_4 {
    public static void main(String[] args) throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        double C, F;
        String str;
        str = br.readLine();
        C = Double.parseDouble(str);
        F=(9 / 5) * C + 32;
        System.out.println("F = "+F);
    }
}
```

【运行结果】

37

F = 69.0

5．已知圆球的体积公式为 $4/3\,\pi\,r^3$，编一程序，输入圆球半径，计算并输出球的体积。

【编程分析】该程序和第 3 题类似，在此不再赘述。

【参考程序】

```
import java.io.*;
public class X2_3_5 {
    public static void main(String[] args) throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        int radius;
        double volume;
        String str;
        System.out.print("Enter the value of radius please: ");
        str = br.readLine();
        radius = Integer.parseInt(str);
        volume=4*Math.PI*Math.pow(radius,3)/3;
        System.out.println("volume = "+volume);
    }
}
```

【运行结果】

```
Enter the value of radius please: 26
volume = 73622.17663932561
```

第 3 章

一、选择题

1．下列（ D ）不属于 Java 语言流程控制结构？

（A）分支语句　　（B）跳转语句　　（C）循环语句　　（D）赋值语句

【解析】Java 语言流程控制结构只包括分支、循环和跳转三种语句。

2．假设 a 是 int 类型的变量，并初始化为 1，则下列（ D ）是合法的条件语句？

（A）if(a){}　　（B）if(a<<=3){}　　（C）if(a=2){}　　（D）if(true){}

【解析】条件语句中的“条件表达式”只能是 boolean 类型，不能是其他类型。

3．下列说法中，不正确的一个是（ C ）。

（A）switch 语句的功能可以由 if…else if 语句来实现

（B）若用于比较的数据类型为 double 型，则不可以用 switch 语句来实现

（C）if …else if 语句的执行效率总是比 switch 语句高

（D）case 子句中可以有多个语句，并且不需要大括号{}括起来

【解析】语句的执行效率高低要根据具体语句来确定，不能简单地说 if …else if 语句的执行效率总是比 switch 语句高。

4．设 a、b 为 long 型变量，x、y 为 float 型变量，ch 为 char 类型变量且它们均已被赋值，则下列语句中正确的是（ B ）。

（A）switch(x+y) {}　　（B）switch(ch+1) {}

（C）switch ch {}　　（D）switch(a+b); {}

【解析】switch 后面的表达式必须放在括号()中，且表达式的类型只能是 byte、short、int 和 char 类型，不能是其他类型。

5．下列循环体执行的次数是（　C　）。

int y=2, x=4;

while(--x != x/y){ }

（A）1　　（B）2　　（C）3　　（D）4

【解析】第 1 次执行 while 时，--x 的值为 3，3/2=1，满足等号两端值不等的条件，执行一次循环体；第 2 次执行 while 时，--x 的值为 2，2/2=1，条件满足，又执行一次循环体；第 3 次执行 while 时，--x 的值为 1，1/2=0，条件满足，再次执行一次循环体；第 4 次执行 while 时，--x 的值为 0，0/2=0，条件不满足满足，结束循环。因此在这个循环语句执行的过程中，循环体共执行了 3 次。

6．下列循环体执行的次数是（　B　）。

int x=10, y=30;

do{ y -= x;　x++; }while(x++<y--);

（A）1　　（B）2　　（C）3　　（D）4

【解析】执行 1 次循环体后，y 的值为 20，x 值为 11，然后执行 while，此时 x++的值为 11，y—的值为 20，因此 x++<y—条件满足，判断完后，x 的值变为 12，y 的值变为 19；接着执行第 2 次循环体，执行完第 2 次循环体后，y 的值为 9，x 值为 13，然后执行 while，此时 x++的值为 13，y—的值为 9，因此 x++<y—条件不满足，结束循环。因此在这个循环语句执行的过程中，循环体共执行了 2 次。

7．已知如下代码：

```
switch(m){
    case 0: System.out.println("Condition 0");
    case 1: System.out.println("Condition 1");
    case 2: System.out.println("Condition 2");
    case 3: System.out.println("Condition 3");break;
    default:System.out.println("Other Condition");
}
```

当 m 的值为（ D　）时，输出“Condition 3”

（A）2　　（B）0、1　　（C）0、1、2　　（D）0、1、2、3

【解析】当 m 的值为 0、1、2、3 时，都能输出“Condition 3”，因为 case 0、case 1、case 2 后面的语句都没有 break，因此可以继续向后执行。

二、填空题

1．跳转语句包括 break、continue、return 和 throw

2．switch 语句先计算 switch 后面的__表达式__的值，再和各__case__语句后的值做比较。

3．if 语句合法的条件值是__boolean__类型。

4．continue 语句必须使用于__循环__语句中。

5．break 语句有两种用途：一种从 switch 语句的分支中跳出，一种是从循环语句内部跳出。

6．do － while 循环首先执行一遍__循环体__，而 while 循环首先判断__表达式的值__。

7．与 C++语言不同，Java 语言不通过__goto__语句实现跳转。

8．每一个 else 子句都必须和它前面的一个距离它最近的__if__子句相对应。

9．在 switch 语句中，完成一个 case 语句块后，若没有通过 break 语句跳出 switch 语句，则会继续执行后面的__case__语句块。

10．在 for 循环语句中可以声明变量，其作用域是__for 循环体__。

三、写出下列程序的运行结果

1．　public class X3_3_1 {

```
        public static void main(String[] args) {
            for(int i=0; i<10; i++){
                if(i==5) break;
                System.out.print(i);
            }
        }
    }
```

【运行结果】01234

【程序解析】本程序主要考查 break 语句的使用。程序中的 for 循环变量 i 是从 0 到 9 进行循环，正常情况下应该输出 0 到 9，但是由于循环体中有“if(i==5) break;”语句，当 i 为 5 时 for 循环就结束了，因此最后的结果是 01234。

2．

```
public class X3_3_2 {
    public static void main(String[] args) {
        int i=5, j=2;
        while(j<i--) j++;
        System.out.print(j);
    }
}
```

【运行结果】4

【程序解析】本程序主要考查 while 语句以及“后缀--”运算符的使用。由于每执行一次 while 判断，i 的值就减 1，每执行 1 次循环体，j 的值就增加 1，而 while 判断总共经历了“2<5”、“3<4”、“4<3”三次，第 3 次时由于条件不满足，还没有执行循环体就结束了循环，因此循环体总共执行了 2 次，j 的值也只加了 2，而其初始值为 2，因此 j 最后的值为 4。

3．

```
public class X3_3_3 {
    public static void main(String[] args) {
        int i=4;
        while(--i>0){       }
        System.out.print(i);
    }
}
```

【运行结果】0

【程序解析】本程序主要考查 while 语句以及“前缀--”运算符的使用。由于 i 的初值为 4，要使“while(--i>0)”结束执行，必须等到 i 的为 0，因此 i 最后的值为 0。

4．

```
public class X3_3_4 {
    public static void main(String[] args) {
        int j=0;
        for(int i=3; i>0; i--){
            j += i;
            int x = 2;
            while(x<j){
                x += 1;
                System.out.print(x);
            }
        }
```

```
        }
    }
```
【运行结果】33453456

【程序解析】本程序主要考查 for 循环和 while 循环嵌套的情况。在 for 循环第 1 次执行到 while 语句时，j 和 x 的值分别为 3 和 2，此时“while(x<j)”条件满足，进入 while 循环体执行，x 值变为 3，然后输出，再进行判断“while(x<j)”，此时条件不满足，结束 while 循环，程序到 for 循环头部的第 3 个表达式“i--”处执行。此时 i 的值变为 2，“i>0”条件满足，进入 for 循环体。在 for 循环第 2 次执行到 while 语句时，j 和 x 的值分别为 5 和 2，此时“while(x<j)”条件满足，进入 while 循环体执行，该循环体共执行 3 次，x 值分别变为 3、4、5，然后输出，在 for 循环第 3 次执行到 while 语句时，j 和 x 的值分别为 6 和 2，此时“while(x<j)”条件满足，进入 while 循环体执行，该循环体共执行 4 次，x 值分别变为 3、4、5、6，然后输出。当程序再次到 for 循环头部的第 3 个表达式“i--”处执行；此时 i 的值变为 0，“i>0”条件不满足，退出 for 循环，结束程序执行。

5．
```
public class X3_3_5 {
    public static void main(String[] args) {
        int i=8, j=2;
        while(j< --i)
            for(int k=0; k<4; k++) j++;
        System.out.print(j);
    }
}
```
【运行结果】6

【程序解析】本程序主要考查 while 循环和 for 循环嵌套执行的情况。第 1 次执行“while(j< --i)”时，j 和—i 的值分别为 2 和 7，条件满足，执行 for 循环，for 循环体共执行了 4 次，j 的值增 4，变为 6；然后又第 2 次执行“while(j< --i)”，此时 j 和—i 的值分别为 6 和 6，条件不满足，结束 while 循环，输出 j 的值。

6．
```
public class X3_3_6 {
    public static void main(String[] args) {
        int a=0, b=1;
        do{
            if(b%2==0)
                a += b;
            b++;
        }while(b <= 100);
        System.out.print(a);
      }
}
```
【运行结果】2550

【程序解析】本程序主要考查 do…while 循环的执行情况。执行循环体之前，a=0，b=1，进入循环体后，由于“b%2==0”的条件不满足，因此直接执行“b++”，b 的值变为 2，满足“while(b <= 100)”条件，再次进入循环体。此时“b%2==0”的条件满足，语句“a+=b”执行，然后执行“b++”，再进行“while(b <= 100)”判断，如此一直执行下去，直至该条件不满足为止，最后输出 a 的值。从循环过程来看，只有 b 为偶数时才加到 a 中去，因此 a 的值应该是“2+4+6+…+100”。

7．
```
public class X3_3_7 {
    public static void main(String[] args) {
        for(int i=1; i<=10; i++){
            if(i<=5) continue;
            System.out.print(i + " ");
        }
    }
}
```
【运行结果】6 7 8 9 10

【程序解析】本程序主要考查 continue 语句的使用情况。由于“if(i<=5) continue;”语句，使得 for 循环变量 i 从 1 到 5 的循环过程中都不能执行“System.out.print(i + " ");”，只有 i 的值从 6 到 10 时才执行该语句，因此输出结果为“6 7 8 9 10”。

8．
```
public class X3_3_8 {
    public static void main(String[] args) {
        char ch='7';
        int r=10;
        switch(ch+1){
            case '7': r += 7;
            case '8': r += 8;
            case '9': r += 9;
            default:
        }
        System.out.print(r);
    }
}
```
【运行结果】27

【程序解析】本程序主要考查 switch…case 语句，由于“ch+1”的值是'8'，因此程序执行了“r += 8;”语句，由于该语句后没有 break，因此又执行了后面的“r += 9;”语句，由于 r 的初值为 10，因此 r 最后的值为 27。

9．
```
public class X3_3_9 {
    public static void main(String[] args) {
lable:  for(int i=0; i<3; i++){
            for(int j=0; j<3; j++){
                if(i==j) continue lable;
                System.out.print(i*3+j+"\t");
            }
            System.out.print("i= "+i);
        }
    }
}
```
【运行结果】3　　6　　7

【程序解析】本程序主要考查 continue lab 语句的运行情况。当程序执行到“continue lable;”语句时，程序直接跳转到外层 for 循环，执行下一次循环。

10．
```
public class X3_3_10 {
```

```
    public static void main(String[] args) {
        int j=0;
a1:   for(int i=3; i>0; i--){
        j += i;
a2:       for(int k=1; k<3;k++){
            j *= k;
            if(i==k)   break a1;
          }
        }
        System.out.println("j= "+j);
    }
}
```

【运行结果】j= 16

【程序解析】本程序主要考查 break lab 语句的执行情况。当程序执行到“break a1;”时，程序流程直接跳出 a1 所在的外层 for 循环，输出 j 的值。

四、编写程序

1．利用 if 语句，根据下列函数编写一个程序，当键盘输入 x 值时，求出并输出 y 的值。

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

【编程分析】本题主要考查 if…else 语句的使用。根据给定的数学算式，只要给出 x 的值，就有对应的 y 算式，因此利用 if…else 语句直接可以实现。

【参考程序】

```
import java.io.*;
public class X3_4_1 {
    public static void main(String[] args) throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        int x, y;
        x = Integer.parseInt(br.readLine());
        if(x<=1) y = x;
        else if(x<10) y = 3*x - 2;
        else y=4*x;
        System.out.println("x = "+x + "\ty = "+y);
    }
}
```

【运行结果】

```
9
x = 9     y = 25
```

2．利用 switch 语句将学生成绩分级，当从键盘中输入学生成绩在 100～90 范围时，输出“优秀”，在 89～80 范围时输出“良好”，在 79～70 范围时输出“中等”，在 69～60 范围时输出“及格”，在 59～0 范围时输出“不及格”，在其他范围时输出“成绩输入有误！”。

【编程分析】本题主要考查 switch 语句的使用。由于要根据一定的数值范围来输出相应的

汉字，而 switch 后面的表达式类型不能是一个范围，因此要把一个整数范围变成一个整数来满足要求。由于 Java 中的 int 和 int 运算的结果还是 int 型，可以把某个给定的程序除 10，如 89/10 和 81/10 的结果都是 8，这样就满足了要求。

【参考程序】

```
import java.io.*;
public class X3_4_2 {
    public static void main(String[] args) throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        int score, k;
        score = Integer.parseInt(br.readLine());
        if(score>100 || score<0){
            System.out.println("输入成绩超出范围！");
            System.exit(1);
        }
        k = score/10;
        switch(k){
            case 10:
            case 9:
                System.out.println("优秀");
                break;
            case 8:
                System.out.println("良好");
                break;
            case 7:
                System.out.println("中等");
                break;
            case 6:
                System.out.println("及格");
                break;
            default:
                    System.out.println("不及格");
        }
    }
}
```

【运行结果】

```
87
良好
```

3．利用 for 循环，计算 1+3+7+……+（2^{20}-1）的和。

【编程分析】本例主要考查利用 for 循环求一个数列的和的编程方法。数列的项数和每一项的值都已知，因此直接利用 for 循环和数学类中的相应方法即可实现。

【参考程序】

```
public class X3_4_3 {
    public static void main(String[] args) {
```

```
        int i, sum=0;
        for(i=1; i<21; i++)
            sum += Math.pow(2,i) - 1;
        System.out.println("sum = " + sum);
    }
}
```

【运行结果】

```
sum = 2097130
```

4．已知 $S=1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots+\frac{1}{n-1}-\frac{1}{n}$，利用 while 循环编程求解 n=100 时的 S 值。

【编程分析】本例主要考查利用 while 循环实现对一个数列进行加、减运算的编程方法。由于项数已经确定，因此数列的首项和末项已知，但是由于加减交替出现，可以利用一个变量 sign 来确定符号，设 sign 初始值为 1，循环一次让它变一次符号，从而实现符号的交替变化。

【参考程序】

```
public class X3_4_4 {
    public static void main(String[] args) {
        int i=1, n = 100;
        double sign = 1, sum = 0;
        while(i<=n){
            double k = sign/i;
            sum += k;
            i++;
            sign = -sign;
        }
        System.out.println("S = " + sum);
    }
}
```

【运行结果】

```
S = 0.688172179310195
```

5．利用 do…while 循环，计算 1!+2!+3! +·····+100!的和。

【编程分析】本例主要考查利用 do…while 循环实现数列相加的程序设计方法。由于每一项都是一个阶乘，所以在循环过程中先求阶乘，然后再相加。

【参考程序】

```
public class X3_4_5 {
    public static void main(String[] args) {
        int i=1;
        long fact = 1, sum = 0;
        do{
            fact *= i;
            sum += fact;
            i++;
        } while(i<=100)    ;
        System.out.println("sum = " + sum);
    }
```

```
    }
```
【运行结果】
```
    sum = 1005876315485501977
```
6．编程序，求 $\sum_{k=1}^{10} k^3$

【编程分析】本例主要考查利用任意一种循环结构实现给定数列求和的程序设计方法。实现方法和前几例类似，在此不再赘述。

【参考程序】
```
    public class X3_4_6 {
        public static void main(String[] args) {
            int k, sum = 0;
            for(k=1;k<=10;k++)
                sum += Math.pow(k,3);
            System.out.println("sum = " + sum);
        }
    }
```
【运行结果】
```
    sum = 3025
```
7．编写打印“九九乘法口诀表”的程序。

【编程分析】本例主要考查利用循环嵌套进行程序设计的方法。“九九乘法口诀表”既涉及到行，又涉及到列，因此需要利用循环嵌套来实现，实现时还需要注意输出格式。

【参考程序】
```
    public class X3_4_7 {
        public static void main(String[] args) {
            int i, j;
            for(i=1;i<10;i++){
                for(j=1;j<=i;j++)
                    System.out.print(i + "x" + j +"=" + i*j + "\t");
                System.out.println("");
            }
        }
    }
```
【运行结果】
```
    1x1=1
    2x1=2    2x2=4
    3x1=3    3x2=6    3x3=9
    4x1=4    4x2=8    4x3=12   4x4=16
    5x1=5    5x2=10   5x3=15   5x4=20   5x5=25
    6x1=6    6x2=12   6x3=18   6x4=24   6x5=30   6x6=36
    7x1=7    7x2=14   7x3=21   7x4=28   7x5=35   7x6=42   7x7=49
    8x1=8    8x2=16   8x3=24   8x4=32   8x5=40   8x6=48   8x7=56   8x8=64
    9x1=9    9x2=18   9x3=27   9x4=36   9x5=45   9x6=54   9x7=63   9x8=72   9x9=81
```
9．水仙花数是指其个位、十位和百位三个数的立方和等于这个三位数本身，求出所有的水

仙花数。

【编程分析】本例主要考查如何利用循环结构将一个多位整数拆分成多个个位数的程序设计方法。在求“水仙花数”过程中，需要将这个百位的数拆分成 3 个个位数。

【参考程序】

```
public class X3_4_9 {
    public static void main(String[] args) {
        for(int i=100;i<=999;i++){
            int k = i;
            int gw = k % 10;
            k /= 10;
            int sw = k % 10;
            k /= 10;
            int bw = k;
            int sum = (int)(Math.pow(gw,3)+Math.pow(sw,3)+Math.pow(bw,3));
            if(i == sum)
                System.out.print(i+ "\t");
        }
    }
}
```

【运行结果】

```
153     370     371     407
```

10．编写一个程序，接受用户输入的两个数据为上、下限，然后输出上、下限之间的所有素数。

【编程分析】本例主要考查如何利用 Java 语言的流程控制语句解决求素数的问题。由于素数是除了能被 1 和它本身整除外，不能被其他任何整数整除的自然数。假设要判断某数 i 是否为素数，可以利用循环语句让该数分别除以 2～i-1 之间的所有数，如果一直不能整除，则 i 是素数，否则不是。如果让 i 分别除以 2～i/2 之间的所有数，如果一直不能整除，则 i 也是素数，因为任何数都不可能被大于它一半的数整除，这样可以减少循环次数，提高程序运行效率。另外，数学原理证明，如果让 i 分别除以 2～$(\text{int})\sqrt{i}$ 之间的所有数，如果一直不能整除，则 i 也是素数，这样就可以大大减少循环次数，提供程序效率。

【参考程序】

```
import java.io.*;
public class X3_4_10 {
    public static void main(String[] args) throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        int top, bottom, i, j;
        System.out.print("请输入上限：");
        top = Integer.parseInt(br.readLine());
        System.out.print("请输入下限：");
        bottom = Integer.parseInt(br.readLine());
        if(top<bottom){
            System.out.println("输入的上、下限不正确！");
```

```
                System.exit(1);
            }
            for(i=bottom; i<=top; i++){
                int k = (int)Math.sqrt(i);
                for(j=2; j<=k; j++){
                    if(i % j == 0) break;
                }
                if(j>k) System.out.print(i + "\t");
            }
            System.out.println();
        }
    }
```

【运行结果】

请输入上限：500
请输入下限：100

101	103	107	109	113	127	131	137	139	149
151	157	163	167	173	179	181	191	193	197
199	211	223	227	229	233	239	241	251	257
263	269	271	277	281	283	293	307	311	313
317	331	337	347	349	353	359	367	373	379
383	389	397	401	409	419	421	431	433	439
443	449	457	461	463	467	479	487	491	499

第 4 章

一、选择题

1．下列哪种类成员修饰符修饰的变量只能在本类中被访问？（ D ）

（A）protected （B）public （C）default （D）private

【解析】只有私有访问权限修饰符才能限制变量只能在本类中被访问。

2．在 Java 语言中，哪一个包中的类是自动导入的？（ A ）

A）java.lang B）java.awt C）java.io D）java.applet

【解析】只有 java.lang 包中的类能够被自动导入。

3．给出下面的程序代码：

```
public class X4_1_3 {
    private float a;
    public static void m ( ){    }
}
```

如何使成员变量 a 被方法 m( )访问（ C ）

（A）将 private float a 改为 protected float a（B）将 private float a 改为 public float a

（C）将 private float a 改为 static float a （D）将 private float a 改为 float a

【解析】static 类型的方法只能访问 static 类型的数据成员。

4．有一个类 B，下面为其构造方法的声明，正确的是（ B ）。

（A）void B(int x) {} （B）B(int x) {}

（C）b(int x) {}　　（D）void b(int x) {}

【解析】构造方法没有类型，且方法名与类名相同。

5．下面关于类的说法，不正确的是（ C ）。

（A）类是同种对象的集合和抽象　　（B）类属于 Java 语言中的复合数据类型

（C）类就是对象　　（D）对象是 Java 语言中的基本结构单位

【解析】类是对象的抽象，对象是类的实例，因此类和对象是有区别的。

6．下面关于方法的说法，不正确的是（ C ）。

（A）Java 中的构造方法名必须和类名相同

（B）方法体是对方法的实现，包括变量声明和合法语句

（C）如果一个类定义了构造方法，也可以用该类的默认构造方法

（D）类的私有方法不能被其他类直接访问

【解析】类中一旦用户定义了构造方法，该类默认的构造方法就不存在了，除非用户再自定义该类的默认构造方法。

7．关于内部类，下列说法不正确的是（ A ）。

（A）内部类不能有自己的成员方法和成员变量

（B）内部类可用 private 或 protected 修饰符修饰

（C）内部类可以作为其他类的成员，而且可访问它所在的类的成员

（D）除 static 内部类外，不能在类内声明 static 成员

【解析】内部类也可以有自己的成员方法和变量。

8．定义外部类时不能用到的关键字是（ C ）。

（A）final　（B）public　（C）protected　（D）abstract

【解析】定义外部类不能使用 protected 修饰符。

9．为 AB 类定义一个无返回值的方法 f，使得使用类名就可以访问该方法，该方法头的形式为（ D ）

（A）abstract void f()　　（B）public void f()

（C）final void f()　　（D）static void f()

【解析】只有 static 类型的方法才可以直接使用类名来引用。

10．定义一个公有 double 型常量 PI，哪一条语句最好？（ B ）

（A）public final double PI;　　（B）public final static double PI=3.14;

（C）public final static double PI;　　（D）public static double PI=3.14;

【解析】用 public final static 来定义常量，既可以节省存储空间，又可以保证数值不会被改变。

二、填空题

1．对象 是对事物的抽象，而 类 是对对象的抽象和归纳。

2．从用户的角度看，Java 源程序中的类分为两种：系统定义的类 和 用户自己定义的类 。

3．一个类主要包含两个要素：数据成员 和 成员方法 。

4．创建包时需要使用关键字 package 。

5．类中的 构造 方法是一个特殊的方法，该方法的方法名和类名相同。

6．如果用户在一个自定义类中未定义该类的构造方法，系统将为这个类定义一个默认构造方法。这个方法没有 形式参数 ，也没有任何 具体语句 ，不能完成任何操作。

7．静态数据成员被保存在类的内存区的 公共存储 单元中，而不是保存在某个对象的内存区中。因此，一个类的任何对象访问它时，存取到的都是 相同 （相同/不同）的数值。

8．静态数据成员既可以通过 对象名 来访问，也可以通过 类名 直接访问它。

9．定义常量时要用关键字 final ，同时需要说明常量的 数据类型 并指出常量的具体值 。

10．方法体内定义变量时，变量前不能加 修饰符 ；局部变量在使用前必须 明确赋值 ，否则编译时会出错；而类变量在使用前可以不用赋值，它们都有一个 默认 的值。

11．static 方法中只能引用 static 类型的数据成员和 static 类型的成员方法；而非 static 类型的方法中既可以引用 static 类型的数据成员和成员方法，也可以引用非 static 类型的数据成员和成员方法。

12．引用 static 类型的方法时，可以使用 类名 做前缀，也可以使用 对象名 做前缀。

13．当程序中需要引用 java.awt.event 包中的类时，导入该包中类的语句为 import java.awt.event.*。

14．定义类时需要 class 关键字，继承类时需要 extends 关键字，实现接口时需要关键字 implements 。

三、编程题

1．编一个程序，程序中包含以下内容：

一个圆类(Circle)，包含：

属性：圆半径 radius；常量：PI。

方法：构造方法；求面积方法 area()；求周长方法：perimeter()。

主类(X4_3_1)，包含：

主方法 main()，在主方法中创建圆类的对象 c1 和 c2 并初始化，c1 的半径为 100，c1 的半径为 200，然后分别显示两个圆的面积和周长。

【编程分析】按照要求创建 Circle 类，其中的半径可以定义为 int 类型，PI 定义为 final static double 类型，求面积和周长的方法都应定义为 double 类型，在构造方法中对 radius 进行初始化。

【参考答案】

```
public class X4_3_1 {
    public static void main(String[] args) {
        Circle c1 = new Circle(100);
        Circle c2 = new Circle(200);
        System.out.println("c1.area() = " +c1.area()+"\tc1.perimenter() = "+c1.perimeter());
        System.out.println("c2.area() = " +c2.area()+"\tc2.perimenter() = "+c2.perimeter());
    }
}
class Circle{
    int radius;
    final static double PI=3.14;
    Circle(int r){
        radius = r;
    }
    double area(){
        return PI*radius*radius;
    }
    double perimeter(){
        return 2*PI*radius;
```

```
        }
    }
```
【运行结果】

c1.area() = 31400.0　　c1.perimenter() = 628.0

c2.area() = 125600.0　　c2.perimenter() = 1256.0

2．编一个程序，程序中包含以下内容：

一个学生类（Student），包含：

属性：学号 s_No，姓名 s_Name，性别 s_Sex，年龄 s_Age。

方法：构造方法，显示学号方法 showNo()，显示姓名方法 showName()，显示性别方法 showSex()，显示年龄方法 showAge()，修改年龄方法 modifyAge()。

主类(X4_3_2)，包含：

主方法 main()，在其中创建两个学生对象 s1 和 s2 并初始化，两个对象的属性自行确定，然后分别显示这两个学生的学号、姓名、性别、年龄，然后修改 s1 的年龄并显示修改后的结果。

【编程分析】按照要求首先编写 Student 类，其中的属性和方法根据实际情况选择相应的类型和权限修饰符，要通过方法来修改年龄，因此 s_Age 属性应该为 private 类型，否则可以直接在主类中进行修改，就没有必要使用方法了。

【参考答案】

```
public class X4_3_2 {
    public static void main(String[] args) {
        Student s1 = new Student(101,"张三","男",18);
        Student s2 = new Student(102,"李四","女",16);
        System.out.println("第 1 个学生的信息为：");
        s1.showNo();
        s1.showName();
        s1.showSex();
        s1.showAge();
        System.out.println("第 2 个学生的信息为：");
        s2.showNo();
        s2.showName();
        s2.showSex();
        s2.showAge();
        System.out.println("修改第 1 个学生的年龄：");
        s1.modifyAge(24);
        System.out.println("第 1 个学生的年龄修改为：");
        s1.showAge();
    }
}
class Student{
    private int s_No;
    private String s_Name;
    private String s_Sex;
    private int s_Age;
    Student(int no, String name, String sex, int age){
```

```
            s_No = no;
            s_Name = name;
            s_Sex = sex;
            s_Age = age;
        }
        void showNo(){
            System.out.println("学号："+s_No);
        }
        void showName(){
            System.out.println("姓名：" + s_Name);
        }
        void showSex(){
            System.out.println("性别：" + s_Sex);
        }
        void showAge(){
            System.out.println("年龄：" + s_Age);
        }
        void modifyAge(int newAge){
            s_Age = newAge;
        }
    }
```

【运行结果】

```
第 1 个学生的信息为：
学号：101
姓名：张三
性别：男
年龄：18
第 2 个学生的信息为：
学号：102
姓名：李四
性别：女
年龄：16
修改第 1 个学生的年龄：
第 1 个学生的年龄修改为：
年龄：24
```

第 5 章

一、选择题

1．已知有下面类的说明：

```
public class X5_1_1 extends x{
    private float f =10.6f;
    int i=16;
    static int si=10;
   public static void main(String[] args) {
```

```
        X5_1_1 x=new   X5_1_1();
      }
   }
```

在 main()方法中，下面哪条语句的用法是正确的？（ A ）

A．x.f　　B．this.si　　C．X5_1_1.i　　D．X5_1_1.f

【解析】由于 x 是在 main 方法内部定义的对象，因此它可以引用类的非 static 类型的属性，因此选 A，而 this 和 super 不能在 main 方法中使用，使用类名只能引用本类的静态属性，因此 B、C、D 不对。

2．下列程序的运行结果是（ C ）。

```
public class X5_1_2 extends x{
   int ab(){
    static int  aa=10;
        aa++;
    System.out.println(aa);
    }
    public static void main(String[] args) {
     X5_1_2 x=new   X5_1_2();
     x.ab();
    }
 }
```

A．10　　B．11

C．编译错误　　D．运行成功，但不输出

【解析】方法体内的局部变量不能使用 static 修饰符。

3．下面关于接口的说法中不正确的是（ C ）。

A．接口中所有的方法都是抽象的

B．接口中所有的方法都是 public 访问权限

C．子接口继承父接口所用的关键字是 implements

D．接口是 Java 中的特殊类，包含常量和抽象方法

【解析】子接口继承父接口所用的关键字也是 extends，只有类实现接口时才使用 implements。

4．区分类中重载方法的依据是（ A ）。

A．形参列表的类型和顺序　　B．不同的形参名称

C．返回值的类型不同　　D．访问权限不同

【解析】形参表的类型和顺序不同时区分重载方法的唯一标志。

5．子类对象能否直接向其父类赋值？父类对象能否向其子类赋值？（ B ）

A．能，能　　B．能，不能　　C．不能，能　　D．不能，不能

【解析】子类对象可以直接赋值给父类对象，而父类对象不能直接赋值给子类对象。

6．Java 语言类间的继承关系是（ A ）。

A．单继承　　B．多重继承　　C．不能继承　　D．不一定

【解析】Java 语言类间的继承关系是单继承，但一个类可以实现多个接口。

7．Java 语言接口间的继承关系是（ B ）。

A．单继承　　B．多重继承　　C．不能继承　　D．不一定

【解析】Java 语言接口间的继承关系允许多重继承。

8．一个类实现接口的情况是（ A ）。

A．一次可以实现多个接口　　B．一次只能实现一个接口

C．不能实现接口　　D．不一定

【解析】Java 语言允许一个类一次实现多个接口。

9．定义外部类的类头时，不可用的关键字是（ C ）。

A．public　　B．final　　C．protected　　D．abstract

【解析】定义外部类时不能使用 protected 关键字。

10．如果局部变量和成员变量同名，如何在局部变量作用域内引用成员变量？（ B ）

A．不能引用，必须改名，使它们的名称不相同

B．在成员变量前加 this，使用 this 访问该成员变量

C．在成员变量前加 super，使用 super 访问该成员变量

D．不影响，系统可以自己区分

【解析】this 可以用来引用本类对象。

11．下面说法不正确的是（ B ）。

A．抽象类既可以做父类，也可以做子类

B．abstract 和 final 能同时修饰一个类

C．抽象类中可以没有抽象方法，有抽象方法的类一定是抽象类或接口

D．声明为 final 类型的方法不能在其子类中重新定义

【解析】abstract 和 final 不能同时修饰一个类，因为 abstract 类需要子类，而 final 类不能有子类。

二、填空题

1．消息就是向对象发出 服务请求 ，是对 数据成员 和 成员方法 的引用。

2．在面向对象系统中，消息分为两类 公有消息 和 私有消息 。

3．在面向对象程序设计中，采用 继承 机制可以有效地组织程序结构。充分利用已有的类来创建更复杂的类，大大提高程序开发的效率，提高代码的复用率，降低维护的工作量。

4．数据成员的隐藏 是指在子类中重新定义一个与父类中已定义的数据成员名完全相同的数据成员。

5．子类可以重新定义与父类同名的成员方法，实现对父类方法的 覆盖 。

6．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 方法名 、返回值类型 和 参数列表 ，否则就不是方法的覆盖，而是子类定义自己特有的方法，与父类的方法无关。

7．this 代表了 当前对象 的一个引用，super 表示的是当前对象的 直接父类对象 的引用。

8．抽象类不能 创建 对象，该工作由抽象类派生的非抽象子类来实现。

9．如果父类中已有同名的 abstract 方法，则子类中就 不能 （能/不能）再有同名的抽象方法。

10．abstract 类中不能有 private 访问权限的数据成员或成员方法。

11．interface 是声明接口的关键字，可以把它看成一个特殊类。接口中的数据成员默认的修饰符是 public static final ，接口中的成员方法默认的修饰符是 public abstract 。

12．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 实现 该接口的所有抽象方法；如果实现某接口的类是 abstract 的抽象类，则它可以 不实现 该接口所有的方法。但是对于这个抽象类任何一个非抽象的子类而言，它们父类所实现的接口中的所有抽象方法以及自身所实现接口中的抽象方法都必须有实在的 方法体 。

13．包的作用有两个，一是 划分类名空间 ，二是 控制类之间的访问 。

14．封装也称 信息隐藏 ，是指类的设计者只为类的使用者提供类的可以访问的部分（包括类的数据成员和成员方法），而把类中的其他成员 隐藏 起来，使用户不能访问的机制。

15．Java 提供了 4 种访问权限来实现封装机制，即 private 、 protected 、 默认 和 public 。

16．Java 中提供两种多态机制， 重载 与 覆盖 。

17．当一个构造方法需要调用另一个构造方法时，可以使用关键字 this ，同时这个调用语句应该是整个构造方法的 第一条 可执行语句。

18．如果子类自己没有构造方法，那么父类也一定 没有 （有/没有）带参的构造方法，此时它将继承父类的 无参构造方法 作为自己的构造方法；如果子类自己定义了构造方法，则在创建新对象时，它将先执行 父类 的构造方法，然后再执行自己的 构造方法 。

19．对于父类的含参数构造方法，子类可以通过在自己的构造方法中使用 super 关键字来调用它，但这个调用语句必须是子类构造方法的 第一条 可执行语句。

20．创建一个名为 myPachkage 的包的语句为 package myPackage; ，该语句应该放在程序第一行 位置。

三、编程题

1．编写一个实现方法重载的程序。

【编程分析】重载的含义就是在一个类中定义多个具有相同方法名，不同参数列表的方法。在下面程序中的类中定义了三个同名方法 area，分别用于求圆、矩形和三角形的面积。三个方法具有不同的参数。

【参考程序】

```
public class X5_3_1{
    public double area(double radius){
        return Math.PI*radius*radius;
    }
    public double area(double width, double height){
        return width*height;
    }
    public double area(double a, double b, double c){
        double s = (a+b+c)/2;
        return Math.sqrt(s*(s-a)*(s-b)*(s-c));
    }
    public static void main(String[] args){

        X5_3_1 shape = new X5_3_1();
        System.out.println("The area of circle is: "+shape.area(10));
        System.out.println("The area of rectangle is: "+shape.area(10,20));
        System.out.println("The area of triangle is: "+shape.area(10,15,20));
    }
}
```

【运行结果】

```
The area of circle is: 314.1592653589793
The area of rectangle is: 200.0
The area of triangle is: 72.61843774138907
```

2．编写一个实现方法覆盖的程序。

【编程分析】方法覆盖是指在子类中重新定义了父类中的方法。本例中在 Shape 类、Circle 类、Cylinder 类中都定义了 area 方法，而且 Cylinder 类继承了 Circle 类、Circle 类继承了 Shape 类，从而实现了 area 方法的覆盖。

【参考程序】

```
abstract class Shape {
    abstract protected double area();
}
```

```
class Circle extends Shape{
    float r;
    public Circle(float a) {
        r=a;
    }
    public double area(){
        System.out.print("Calculate the area of circle:    ");
        return Math.PI*r*r;
    }
}
class Cylinder extends Circle {
    float h;
    public Cylinder(float a,float b) {
        super(a);
        h=b;
    }
    public double area() {
        System.out.print("Calculate the area of cylinder:    ");
        return 2*Math.PI*r*r+2*Math.PI*r*h;
    }
 }
public class EX5_3_2 {
    public static void main(String args[]) {
        Circle cl=new Circle(3);
        Cylinder cd=new Cylinder(2,5);
        System.out.println(cl.area());
        System.out.println(cd.area());
    }
}
```

【运行结果】

```
Calculate the area of circle:    28.274333882308138
Calculate the area of cylinder:    87.96459430051421
```

3．编写一个实现数据成员隐藏的程序。

【编程分析】数据成员的隐藏是指子类和父类都有一个相同的数据成员，当子类对象调用该数据成员时，默认调用子类的该成员，而不是父类的该成员，相当于隐藏了父类的该成员。本例中 Hide 类中定义了数据成员 x，该类的子类 X_5_3_3 中也定义了数据成员 x，当子类对象调用 x 时，调用的是子类的 x，而不是父类的 x。

【参考程序】

```
public class X5_3_3 extends Hide {
    int x = 200;
    public static void main(String args[]) {
        X5_3_3 obj = new X5_3_3 ();
        System.out.println("x = "+obj.x);  // 数据成员的隐藏
    }
```

```
    }
    class Hide{
        int x = 100;
    }
```

【运行结果】

```
    x = 200
```

4．编写一个使用 this 和 super 关键字的程序。

【编程分析】this 关键字表示当前对象本身，而 super 关键字则是当前对象的直接父类对象。本例程序中定义了两个类 X1 和 X2，其中都定义了成员变量 x、y。此时就需要区分父类与子类的成员变量，采用 super 引用父类的成员变量和方法。

注意：父类和子类中都定义了 show()方法，如果没有特别指定，则是调用当前类的方法。

【参考程序】

```
    public class X5_3_4{
        public static void main(String[] args){
            X2 x = new X2(5);
            x.setY();
            x.show();
            System.out.println("super.y="+x.getY());     // 输出父类数据成员 y
            System.out.println("this.y="+x.y);           // 输出子类数据成员 y
        }
    }
    class X1{
        int x,y;
        X1(int i){
            x=i;
            y=x*2;
        }
        int getY(){
            return y;
        }
        void show(){
            System.out.println("x="+x+" y="+y);
        }
    }
    class X2 extends X1{
        int x,y;
        X2(int j){
            super(3);            // 利用 super 来调用父类的构造方法
            this.x=j;            // 利用 this 来调用本类的数据成员 x
        }
        void show(){
            System.out.println("super.x="+super.x+" this.x="+x);  // 输出父类和子类的属性
    x
        }
```

```
        void setY(){
            y=super.x+this.x;   // 引用父类和子类中的数据成员 x
        }
    }
```

【运行结果】

```
super.x=3 this.x=5
super.y=6
this.y=8
```

5．编写一个人类 Person，其中包含姓名、性别和年龄的属性，包含构造方法以及显示姓名、性别和年龄的方法。再编写一个学生类 Student，它继承 Person 类，其中包含学号属性，包含构造方法以及显示学号的方法。最后编写一个主类 X5_3_5，包含 main()方法，在 main()方法中定义两个学生 s1 和 s2 并给他们赋值，最后显示他们的学号、姓名、性别以及年龄。

【编程分析】本题主要考察类的继承问题。

第一步：定义 Person 类。

第二步：定义 Student 类，该类继承 Person 类。

第三步：定义主类。

【参考程序】

```
public class X5_3_5 {
      public static void main(String[] args) {
      Student s1=new Student("Zhangsan","Male",20,"102A");
      Student s2=new Student("Lisi","Female",18,"108S");
      s1.show();
      s1.showID();
      s2.show();
      s2.showID();
      }
}
class Person{
    String name;
    String sex;
    int age;
    public Person(String n,String s,int a){
     name = n;
     sex = s;
     age = a;
    }
    void show(){
     System.out.println("name: "+name);
     System.out.println("sex: "+sex);
     System.out.println("age: "+age);
    }
}
class Student extends Person{
```

```
    String sID;
    public Student(String n,String s,int a,String sid){
     super(n,s,a);
     sID = sid;
    }
    void showID(){
     System.out.println("sID: "+sID);
    }
}
```

【运行结果】

```
name: Zhangsan
sex: Male
age: 20
sID: 102A
name: Lisi
sex: Female
age: 18
sID: 108S
```

6．编一个程序，包含以下文件。

（1）Shape.java 文件，在该文件中定义接口 Shape，该接口在 shape 包中。

属性：PI。

方法：求面积的方法 area()。

（2）Circle.java 文件，在该文件中定义圆类 Circle，该类在 circle 包中，实现 Shape 接口。

属性：圆半径 radius。

方法：构造方法；实现接口中求面积方法 area()；求周长方法 perimeter()。

（3）“Cylinder.java”文件，在该文件中定义圆柱体类 Cylinder，该类口在 cylinder 包中，继承圆类。

属性：圆柱体高度 height。

方法：构造方法；求表面积方法 area()；求体积方法 volume()。

（4）X5_3_6.java 文件，在该文件中定义主类 X5_3_6，该类在默认包中，其中包含主方法 main()，在主方法中创建两个圆类对象 cir1 和 cir2，具体尺寸自己确定，并显示圆的面积和周长；再创建两个圆柱体类的对象 cy1 和 cy2，具体尺寸自己确定，然后分别显示圆柱体 cy1 和 cy2 的底圆的面积和周长以及它们各自的体积和表面积。

【编程分析】本题主要考察接口、包、继承、封装等问题。编程步骤如下：

第一步：首先创建 p1 包，在其中创建 Shape 接口

```
// Shape.java 文件
package p1;    // 创建 p1 包
public interface Shape{        // 定义 Shape 接口
      …
}
```

第二步：创建 Circle 类和 Cylinder 类，它们都定义在 p2 包中。

```
// Circle.java 文件
package p2;    // 创建 p2 包
import p1.*;
```

```
public class Circle implements Shape{  // 定义实现 Shape 接口的 Circle 类
    …
}
// Cylinder.java 文件
package p2;
public class Cylinder extends Circle{   // 创建继承 Circle 类的 Cylinder 类
    …
}
```

第三步：创建主类，在其中的 main()方法中创建对象，实现相应的功能。

```
// X5_3_6.java 文件
package p3;
import p2.*;
public class X5_3_6 {       // 定义主类
    public static void main(String[] args) {
        …
    }
}
```

【参考程序】

```
// X5_3_6.java 文件
package p3;
import p2.*;
public class X5_3_6 {       // 定义主类
    public static void main(String[] args) {
        Circle cir1 = new Circle(120.5);
        Circle cir2 = new Circle(183.8);
        System.out.println("cir1.area: "+cir1.area());
        System.out.println("cir1.perimeter: "+cir1.perimeter());
        System.out.println("cir2.area: "+cir2.area());
        System.out.println("cir2.perimeter: "+cir2.perimeter());
        Cylinder cy1 = new Cylinder(27.3,32.7);
        Cylinder cy2 = new Cylinder(133.5,155.8);
        System.out.println("cy1.area: "+cy1.area());
        System.out.println("cy1.volume: "+cy1.volume());
        System.out.println("cy2.area: "+cy2.area());
        System.out.println("cy2.volume: "+cy2.volume());
    }
}
// Shape.java 文件
package p1;   // 创建 p1 包
public interface Shape{        // 定义 Shape 接口
    double PI=Math.PI;
    double area();        // 求面积方法
}
// Circle.java 文件
```

```
package p2;    // 创建 p2 包
import p1.*;
public class Circle implements Shape{  // 定义实现 Shape 接口的 Circle 类
    double radius; // 半径
    public Circle(double r){
        radius = r;
    }
    public double area(){     // 实现 Shape 接口中的方法（这是必须的）
        return PI*radius*radius;
    }
    public double perimeter(){     // 定义求圆周长的方法
        return 2*PI*radius;
    }
}
// Cylinder.java 文件
package p2;
public class Cylinder extends Circle{    // 创建继承 Circle 类的 Cylinder 类
    double height;
    public Cylinder(double r,double h){
        super(r);
        height = h;
    }
    public double area(){
        return 2*PI*radius*radius+2*PI*radius*height;
    }
    public double volume(){
        return PI*radius*radius*height;
    }
}
```

【运行结果】

```
cir1.area: 45616.710728287195
cir1.perimeter: 757.1238295151402
cir2.area: 106130.66532433797
cir2.perimeter: 1154.849459459608
cy1.area: 10291.857533160162
cy1.volume: 76563.70115356173
cy2.area: 242666.35550050176
cy2.volume: 8723280.89865466
```

第 6 章

一、选择题

1．给出下面程序代码：

```
byte[] a1, a2[];
```

```
byte a3[][];
byte[][] a4;
```

下列数组操作语句中哪一个是不正确的？（ A ）

A．a2 = a1　　B．a2 = a3　　C．a2 = a4　　D．a3 = a4

【解析】只有维数相同的数组才能相互赋值。

2．关于数组，下列说法中不正确的是（ C ）。

A．数组是最简单的复合数据类型，是一系列数据的集合

B．数组元素可以是基本数据类型、对象或其他数组

C．定义数组时必须分配内存

D．一个数组中所有元素都必须具有相同的数据类型

【解析】数组元素可以是基本数据类型、对象或其他数组。

3．设有下列数组定义语句：

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

则对此语句的叙述错误的是（ C ）。

A．定义了一个名为 a 的一维数组　　B．a 数组有 3 个元素

C．a 数组元素的下标为 1～3　　D．数组中每个元素的类型都是整数

【解析】数组元素的下标是从 0 开始的。

4．执行语句：int[] x = new int[20];后，下面哪个说法是正确的？（ C ）

A．x[19]为空　　B．x[19]未定义　　C．x[19]为 0　　D．x[0]为空

【解析】此语句定义了 x 数组后，x[0]~x[19]的值全部为 0。

5．下面代码运行后的输出结果为（ A ）。

```
public class X6_1_5 {
    public static void main(String[] args) {
        AB aa = new AB();
        AB bb;
        bb = aa;
        System.out.println(bb.equals(aa));
  }
 }
 class AB{ int x = 100; }
```

A．true　　B．false　　C．编译错误　　D．100

【解析】同一个类的两个对象可以相互赋值，赋值后两个对象具有相同的存储空间，因此是相同的。

6．已知有定义：String s="I love"，下面哪个表达式正确？（ A ）

A．s += "you";　　B．char c = s[1];

C．int len = s.length;　　D．String s = s.toLowerCase();

【解析】字符串对象可以执行“+=”运算，但不能用 s[1]这种方式取其中的字符，也不能用 length 求它的长度，可以用 length()求其长度，因此 B、C 不正确，不能再次定义 s 字符串，因此 D 不正确。

二、填空题

1．Object 类 是所有类的直接或间接父类，它在 java.lang 包中。

2．System 类是一个功能强大、非常有用的特殊的类，它提供了 标准输入/输出 、 运行时 系统信息等重要工具。这个类不能 实例化 ，即不能创建 System 类的对象，所以它所有的属性和方法都是 static 类型，引用时以类名 System 为前缀即可。

3．Applet 由浏览器自动调用的主要方法 init ， start ， stop 和 destroy 分别对应了 Applet 从初始化、启动、暂停到消亡的生命周期的各个阶段 。

4．数组是一种 复合 数据类型，在 Java 中，数组是作为 对象 来处理的。数组是有限

元素的有序集合，数组中的元素具有相同的__数据类型__，并可用统一的__数组名__和__下标__来唯一确定其元素。

5．在数组定义语句中，如果[]在数据类型和变量名之间时，[]之后定义的所有变量都是__数组__类型，当[]在变量名之后时，只有[]之前的变量是__数组__类型，之后没有[]的则不是数组类型。

6．数组初始化包括__静态__初始化和__动态__初始化两种方式。

7．利用__System__类中的__arraycopy()__方法可以实现数组元素的复制；利用__Arrays__类中的__sort()__和__binarySearch()__方法可以实现对数组元素的排序、查找等操作。

8．Java 语言提供了两种具有不同操作方式的字符串类：__String__类和__StringBuffer__类。它们都是__java.lang.Object__的子类。

三、写出下列程序的运行结果

1．
```
public class X6_3_1 {
      public static void main(String[] args) {
         int a[]={12,39,26,41,55,63,72,40,83,95};
         int i1=0,i2=0;
         for(int i=0;i<a.length;i++)
             if(a[i] %2 == 1) i1++;
             else i2++;
         System.out.println(i1+"\t"+i2);
   }
```
【运行结果】
```
6      4
```

2．
```
public class X6_3_2 {
      public static void main(String[] args) {
         int a[]={36,25,48,14,55,40,32,66};
         int b1,b2;
         b1=b2=a[0];
         for(int i=1;i<a.length;i++)
             if ( a[i] >b1 ){
                 if ( b1 >b2 ) b2=b1;
                 b1=a[i];
             }
         System.out.println(b1+"\t"+b2);
      }
   }
```
【运行结果】
```
66     55
```

3．
```
public class X6_3_3 {
      public static void main(String[] args) {
         int a[]={36,25,48,14,55,40,32,66 };
         int b1,b2;
         b1=b2=a[0];
         for (int i=1;i<a.length;i++)
           if ( a[i]<b1 )  {
             if ( b1<b2 )  b2=b1;
             b1=a[i];
         }
         System.out.println(b1+"\t"+b2);
      }
   }
```
【运行结果】
```
14     25
```

4．
```
public class X6_3_4 {
```

```
        public static void main(String[] args) {
          String str = "abcdabcabfgacd";
          char[] a =str.toCharArray();
          int i1 =0, i2=0, i;
          for(i=0;i<a.length;i++) {
              if(a[i] == 'a' )  i1++;
              if(a[i] == 'b' )  i2++;
          }
          System.out.println(i1+"\t"+i2);
       }
    }
```

【运行结果】

```
4       3
```

5.
```
public class X6_3_5 {
      public static void main(String[] args) {
          String str ="abcdabcabdaeff";
          char[] a = str.toCharArray();
          int b[] = new int[5],i;
          for(i=0;i<a.length;i++) {
             switch (a[i]) {
                case 'a': b[0] ++; break;
                case 'b': b[1] ++; break;
                case 'c': b[2] ++; break;
                case 'd': b[3] ++; break;
                default : b[4] ++;
             }
          }
          for(i =0; i<5; i++)
              System.out.print(b[i]+"\t");
          System.out.println();
       }
    }
```

【运行结果】

```
4       3       2       2       3
```

6.
```
public class X6_3_6 {
      public static void main(String[] args) {
          int a[] = {76,83,54,62,40,75,90,92,77,84};
          int b[] = {60,70,90,101};
          int c[]=new int[4],i;
          for (i=0; i<a.length; i++) {
              int j =0;
              while (a[i] >= b[j] ) j ++;
              c[j] ++;
          }
          for (i=0; i<4; i++)
              System.out.print(c[i]+"\t");
          System.out.println();
       }
}
```

【运行结果】

```
2       1       5       2
```

7.
```
public class X6_3_7 {
       public static void main(String[] args) {
          int a[][] = {{1,2,7,8},{5,6,11,12},{9,10,3,4}};
          int m = a[0][0];
          int ii = 0, jj = 0;
          for (int i=0;i<a.length;i++)
```

```
            for(int j=0;j<a[i].length;j++)
                    if ( a[i][j]>m ){
                        m =a[i][j];
                        ii =i;
                        jj =j;
                    }
            System.out.println(ii+"\t"+jj+"\t"+a[ii][jj]);
        }
      }
```

【运行结果】

```
1       3       12
```

8.

```
public class X6_3_8 {
    public static void main(String[] args) {
        String[] a = {"student" ,"worker" ,"cadre" ,"soldier" ,"peasant" };
        String s1,s2;
        s1 = s2 = a[0];
        for( int i = 1; i<a.length;i ++) {
            if (a[i].compareTo(s1)>0) s1=a[i];
            if (a[i].compareTo(s2)<0) s2=a[i];
        }
        System.out.println(s1+"\t"+s2);
    }
}
```

【运行结果】

```
worker  cadre
```

四、编程题

1．有一个数列，它的第一项为 0，第二项为 1，以后每一项都是它的前两项之和，试产生该数列的前 20 项，并按逆序显示出来。

【编程分析】本例由于涉及到 20 项数据的存储，因此可以利用数组来实现。由于数列的各项之间存在一定的关系，可以利用前两项来计算后面项。

【参考答案】

```
public class X6_4_1 {
    public static void main(String[] args) {
    int[] a = new int[20];
    a[0]=0;
    a[1]=1;
    int i;
    for(i=2;i<20;i++){
        a[i]=a[i-1]+a[i-2];        // 求其余 18 项
    }
    for(i=0;i<20;i++){             // 每一行显示 5 个元素
        if(i%5 == 0)
            System.out.println();
        System.out.print(a[i]+"\t");
    }
    System.out.println();
   }
 }
```

【运行结果】

```
0       1       1       2       3
```

5 8 13 21 34
55 89 144 233 377
610 987 1597 2584 4181

2．首先让计算机随机产生出 10 个两位正整数，然后按照从小到大的次序显示出来。

【编程分析】首先利用 Math.random()方法，让计算机随机产生 10 个两位数的正整数，然后编写一个排序方法，实现对数组的排序。（也可以利用 java.utitl.Arrays 类提供的排序方法排序）

【参考答案】

```
public class X6_4_2 {
    public static void main(String[] args) {
        int[] a = new int[10];
        int i;
        for(i=0;i<10;i++){
            a[i]=10+(int)(90*Math.random());        // 产生 10 个两位数的随机数
        }
        System.out.println("随机产生的数据为：");
        for(i=0;i<10;i++){
            System.out.print(a[i]+"\t");
        }
        System.out.println("排序后的数据为：");
        selectSort(a);          // 排序方法的调用
        for(i=0;i<10;i++){
            System.out.print(a[i]+"\t");
        }
        System.out.println();
    }
    static void selectSort(int[] aa){      // 选择排序方法的定义
        int i,j,k;
        for(i=1;i<aa.length;i++){
            k=i-1;
            for(j=i;j<aa.length;j++){
                if(aa[j]<aa[k]) k=j;
            }
            int x = aa[i-1];
            aa[i-1]=aa[k];
            aa[k]=x;
        }
    }
}
```

【运行结果】

随机产生的数据为：
33 67 47 31 19 11 77 65 88 84
排序后的数据为：
11 19 31 33 47 65 67 77 84 88

3．从键盘上输入 4 行 4 列的一个实数矩阵到一个二维数组中，然后求出主对角线上元素之乘积以及副对角线上元素之乘积。

【编程分析】本例主要考察二维数组及其动态初始化的方法。

第一步：创建输入流对象。

```
InputStreamReader isr = new InputStreamReader(System.in);  // 创建输入流对象
BufferedReader br=new BufferedReader(isr);
```

第二步：求主、付对角线乘积

```
for(i=0;i<M;i++){
    main_product *= a[i][i];       // 计算主对角线元素的乘积
    vice_product *= a[i][M-i-1]; // 计算付对角线元素的乘积
}
```

【参考程序】

```
import java.io.*;
public class X6_4_3 {
    public static void main(String[] args) throws IOException{
    final int M = 4;
    double[][] a = new double[M][M];
    int i,j;
    double main_product=1.0,vice_product=1.0;
    InputStreamReader isr = new InputStreamReader(System.in);  // 创建输入流对象
    BufferedReader br=new BufferedReader(isr);
    for(i=0;i<M;i++)
        for(j=0;j<M;j++)
            a[i][j]=Double.parseDouble(br.readLine());
    for(i=0;i<M;i++){
        main_product *= a[i][i];       // 计算主对角线元素的乘积
        vice_product *= a[i][M-i-1]; // 计算付对角线元素的乘积
    }
    System.out.println("主对角线乘积为："+main_product);
    System.out.println("付对角线乘积为："+vice_product);
    }
  }
```

【运行结果】

```
1
2
3
4
5
6
7
8
9
10
11
```

12
13
14
15
16
主对角线乘积为：1056.0
付对角线乘积为：3640.0

4．已知一个数值矩阵 A 为$\begin{bmatrix}3&0&4&5\\6&2&1&7\\4&1&5&8\end{bmatrix}$，另一个矩阵 B 为 A 为$\begin{bmatrix}1&4&0&3\\2&5&1&6\\0&7&4&4\\9&3&6&0\end{bmatrix}$，求出 A 与 B 的乘积矩阵 C[3][4]并输出出来，其中 C 中的每个元素 C[i][j]等于$\sum A[i][k]*B[k][j]$。

【编程分析】本例主要考察二维数组及其静态初始化的方法，以及如何利用二维数组实现矩阵的乘积。主要步骤如下：

第一步：定义三个二维数组 A、B、C，其中 A、B 为题目中给的数组，在定义的同时进行静态初始化，C 数组为 A、B 的乘积，其行数为 A 矩阵的行数、列数为 B 矩阵的列数。

第二步：利用公式 C[i][j]=$\sum A[i][k]*B[k][j]$ 求数组 C。

【参考程序】

```
import java.io.*;
public class X6_4_4 {
    public static void main(String[] args) throws IOException{
    final int M =3, N=4, P=4;
    int i,j,k;
    int[][] a = {{3,0,4,5},{6,2,1,7},{4,1,5,8}};
    int[][] b = {{1,4,0,3},{2,5,1,6},{0,7,4,4},{9,3,6,0}};
    int[][] c = new int[M][N];
    for(i=0;i<M;i++)
        for(j=0;j<N;j++)
            for(k=0;k<P;k++)
                c[i][j] += a[i][k]*b[k][j];
    for(i=0;i<M;i++){
        for(j=0;j<N;j++)
            System.out.print(c[i][j]+"\t");
        System.out.println();
    }
}
 }
```

【运行结果】

```
48      55      46      25
73      62      48      34
78      80      69      38
```

5．从键盘上输入一个字符串，试分别统计出该字符串中所有数字、大写英文字母、小写英文字母以及其他字符的个数并分别输出这些字符。

【编程分析】本题主要考察字符串的输入及字符串方法的应用。

第一步：建立输入流对象，实现从键盘输入字符串。

第二步：利用循环语句及字符串类中的方法 charAt()，对输入字符串中的每个字符进行判断，并统计出各类字符的个数。

【参考程序】

```
import java.io.*;
public class X6_4_5 {
    public static void main(String args[]) throws IOException{
        InputStreamReader isr= new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        String str = br.readLine();
        int i, ditNo=0, upCharNo=0, loCharNo=0, otherCharNo=0;
        for(i=0;i<str.length();i++){
            if(str.charAt(i)<='9' && str.charAt(i)>='0')
                ditNo++;
            else if(str.charAt(i)<='Z' && str.charAt(i)>='A')
                upCharNo++;
            else if(str.charAt(i)<='z' && str.charAt(i)>='a')
                loCharNo++;
            else
                otherCharNo++;
        }
        System.out.println("ditNo = "+ditNo+"\t"+"upCharNo = "+upCharNo);
        System.out.println("loCharNo="+loCharNo+"\t"+"otherCharNo = "+otherCharNo);
    }
}
```

【运行结果】

```
abcABCDE1234<>><?..dF5
ditNo = 5           upCharNo = 6
loCharNo = 4       otherCharNo = 7
```

6．从键盘上输入一个字符串，利用字符串类提供的方法将大写字母转变为小写字母，小写字母转变为大写字母，再将前后字符对换，然后输出最后结果。

【编程分析】本题主要考察 StringBuffer 类及其方法的应用。

第一步：创建输入流对象。

第二步：创建 StringBuffer 类对象。

第三步：利用 StringBuffer 类中的方法，实现将大写字母转变为小写字母，小写字母转变为大写字母，以及将前后字符对换操作。

【参考程序】

```
import java.io.*;
public class X6_4_6 {
    public static void main(String args[]) throws IOException{
        InputStreamReader isr= new InputStreamReader(System.in);
```

```
            BufferedReader br = new BufferedReader(isr);
            StringBuffer strb = new StringBuffer(br.readLine());
            int i;
            for(i=0; i<strb.length(); i++){
                char ch = strb.charAt(i);
                if(ch<='Z' && ch>='A')
                    strb.setCharAt(i,(char)(ch+32));
                if(ch<='z' && ch>='a')
                    strb.setCharAt(i,(char)(ch-32));
            }
            strb.reverse();
            System.out.println(strb);
        }
    }
```

【运行结果】

abc123DEFG

gfed321CBA

第 7 章

一、选择题

1．关于异常的含义，下列描述中最正确的一个是（ D ）。

A．程序编译错误　　B．程序语法错误

C．程序自定义的异常事件　　D．程序编译或运行时发生的异常事件

【解析】异常就是程序编译或运行时发生的异常事件。

2．自定义异常时，可以通过对下列哪一项进行继承？（ C ）

A．Error 类　　B．Applet 类

C．Exception 类及其子类　　D．AssertionError 类

【解析】自定义异常类时，该类必须继承 Exception 类及其子类。

3．对应 try 和 catch 子句的排列方式，下列哪一项是正确的？（ A ）

A．子类异常在前，父类异常在后　　B．父类异常在前，子类异常在后

C．只能有子类异常　　D．父类和子类不能同时出现在 try 语句块中

【解析】对应 try 和 catch 子句的排列方式，要求子类异常（范围小的异常）在前，父类异常（范围大的异常）在后。

4．运行下面程序时，会产生什么异常？（ A ）

```
public class X7_1_4 {
    public static void main(String[] args) {
        int x = 0;
        int y = 5/x;
        int[] z = {1,2,3,4};
        int p = z[4];
  }
}
```

A．ArithmeticException　　B．NumberFormatException

C．ArrayIndexOutOfBoundsException　　D．IOException

【解析】当程序执行到“int y = 5/x”语句时，发生异常，程序中止执行，因此发生

ArithmeticException 异常。

5．运行下面程序时，会产生什么异常？（ C ）

```
public class X7_1_5 {
    public static void main(String[] args) {
        int[] z = {1,2,3,4};
        int p = z[4];
        int x = 0;
        int y = 5/x;
    }
}
```

A．ArithmeticException　　B．NumberFormatException

C．ArrayIndexOutOfBoundsException　　D．IOException

【解析】当程序执行到“int p = z[4]”语句时，发生异常，程序中止执行，因此发生 ArrayIndexOutOfBoundsException 异常。

6．下列程序执行的结果是（ B ）。

```
public class X7_1_6 {
    public static void main(String[] args) {
        try{
            return;
        }
        finally{
            System.out.println("Finally");
        }
    }
}
```

A．程序正常运行，但不输出任何结果　　B．程序正常运行，并输出 Finally

C．编译通过，但运行时出现异常　　D．因为没有 catch 子句，因此不能通过编译

【解析】在执行 try-catch-finally 语句块时，最后必须执行 finally 语句块中的内容，而本程序没有异常发生，因此程序正常运行，并输出 Finally。

7．下列代码中给出正确的在方法体内抛出异常的是（ B ）。

A．new throw Exception(" ");　　B．throw new Exception(" ");

C．throws IOException();　　D．throws IOException;

【解析】在方法体内抛出异常时只能使用 throw，而不能使用 throws，另外，“new Exception(" ")”是创建一个异常，因此 B 是正确的。

8．下列描述了 Java 语言通过面相对象的方法进行异常处理的好处，请选出不在这些好处范围之内的一项（ C ）

A．把各种不同的异常事件进行分类，体现了良好的继承性

B．把错误处理代码从常规代码中分离出来

C．可以利用异常处理机制代替传统的控制流程

D．这种机制对具有动态运行特性的复杂程序提供了强有力的支持

【解析】异常处理机制不能代替传统的流程控制。

二、填空题

1．异常是在程序编译或运行中所发生的可预料或不可预料的异常事件，出现在编译阶段的异常，称之为 编译时异常 ，出现在运行阶段的异常，称之为 运行时异常 。

2．根据异常的来源，可以把异常分为两种类型： 系统定义的运行时异常 和 用户自定义异常 。

3．所有的 Java 异常类都是系统类库中的 Exception 类的子类。

4．抛出异常分为 由系统自动抛出异常 、 通过 throw 抛出异常 以及 通过 throws 抛出异常 三种情况。

5．Java 语言为我们提供了 try…catch 语句和 try…catch…finally 语句捕捉并处理异常。

6．一个 try 块后面可能会跟着若干个 catch 块，每个 catch 块都有一个异常类名作为参数。

7．如果 try 语句块产生的异常对象被第一个 catch 块所接收，则程序的流程将 转向第一个 catch 块 ，catch 语句块执行完毕后就 退出当前方法 ，try 块中尚未执行的语句和其他的 catch 块将被 忽略 ；如果 try 语句块产生的异常对象与第一个 catch 块不匹配，系统将自动转到 第二个 catch 块 进行匹配。

8．由于异常对象与 catch 块的匹配是按照 catch 块的 先后 顺序进行的，所以在处理多异常时应注意认真设计各 catch 块的排列顺序。

9．throws 语句抛出的异常实际上是由 throws 语句修饰的方法内部的 throw 语句抛出的，使用 throws 的主要目的是为了 通知所有预调用此方法的方法 。

三、编程题

1．编写一个系统自动抛出的、系统自行处理的数组大小为负数的程序。

【编程分析】

当编写的程序中没有处理异常的语句时，系统会自动抛出异常，并自行进行处理。

【参考程序】

```
public class X7_3_1 {
    public static void main(String[] args){
        int[] a = new int[-5];
        for(int i=0; i<a.length; i++){
            a[i] = 10 + i;
        }
    }
}
```

【运行结果】

```
Exception in thread "main" java.lang.NegativeArraySizeException
        at X7_3_1.main(X7_3_1.java:3)
```

2．编写一个由 throw 抛出的、系统自行处理的数组下标越界的程序。

【编程分析】

当由 throw 抛出异常后，如果程序本身不进行异常处理，Java 系统将采用默认的处理方式进行处理。

【参考程序】

```
import java.io.*;
public class X7_3_2 {
    public static void main(String[] args)throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        int[] a = new int[5];
        int n = Integer.parseInt(br.readLine());
        if(n>5)    // 当输入的 n 值大于 5 时将由 throw 抛出异常
            throw new ArrayIndexOutOfBoundsException();
        System.out.println("程序继续执行");
        for(int i=0; i<n; i++){
            a[i] = 10 + i;
            System.out.print(a[i]+"\t");
```

```
            }
            System.out.println();
        }
    }
```
【运行结果】
```
8
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
        at C1.main(C1.java:9)
```
3．编写一个系统自动抛出的、由 try－catch 捕捉处理的分母为 0 以及数组下标越界的程序。

【编程分析】

当在 try 语句块中出现分母为 0 的情况时，如果在 catch 参数中有 ArithmeticException 对象，则就能捕捉到该异常并进行处理；同样，当在 try 语句块中出现分母为数组下标越界的情况时，如果在 catch 参数中有 ArrayIndexOutOfBoundsException 对象，则就能捕捉到该异常并进行处理。

【参考程序】
```
import java.io.*;
public class X7_3_3 {
    public static void main(String args[]) throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        System.out.println("请输入两个整数：");
        int a = Integer.parseInt( br.readLine());
        int b = Integer.parseInt( br.readLine());
        try{ // 当输入的 b 为 0 时，就会出现算术异常
            System.out.println(a/b);
        }
        catch(ArithmeticException e){     // 捕捉算术异常并进行处理
            System.out.println("出现被 0 除的情况！");
        }
        int c[] =new int[4], sum = 0;
        try{ // 当出现数组下标越界时就会抛出异常
            for(int i = 0; i<5; i++)    sum += c[i];
            System.out.println( " sum = " + sum);
        }
        catch(ArrayIndexOutOfBoundsException e){// 捕捉数组下标越界异常并处理
            System.out.println("数组下标越界！");
        }
    }
}
```
【运行结果】
```
请输入两个整数：
20
0
出现被 0 除的情况！
```

数组下标越界！

4．编写一个由 throw 抛出的、由 try－catch 捕捉处理的分母为 0 以及数组下标越界的程序。

【编程分析】

当在程序出现异常之前利用 throw 语句来抛出异常，可以做到防患于未然，提前进行异常处理，将由被动处理异常转变为主动防止异常发生。

【参考程序】

```
import java.io.*;
public class X7_3_4 {
    public static void main(String args[]) throws IOException{
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        System.out.println("请输入两个整数：");
        int a = Integer.parseInt( br.readLine());
        int b = Integer.parseInt( br.readLine());
        try{
            if(b==0)
                throw new ArithmeticException("抛出算术异常");
            System.out.println(a/b);
        }
        catch(ArithmeticException e){
            e.printStackTrace();
            System.out.println("出现被 0 除的情况！");
        }
        int c[] ={1, 2, 3, 4}, sum = 0;
        try{
            for(int i = 0; i<5; i++)    {
                if(i >= 4)
                    throw new ArrayIndexOutOfBoundsException("抛出数组下标越界
异常");
                sum += c[i];
                System.out.println( " sum = " + sum);
            }
        }
        catch(ArrayIndexOutOfBoundsException e){
            e.printStackTrace();
            System.out.println("数组下标越界！");
        }
    }
}
```

【运行结果】

```
请输入两个整数：
10
0
java.lang.ArithmeticException: 抛出算术异常
```

```
        at X7_3_4.main(C9_4.java:11)
出现被 0 除的情况!
 sum = 1
 sum = 3
 sum = 6
 sum = 10
java.lang.ArrayIndexOutOfBoundsException: 抛出数组下标越界异常
        at X7_3_4.main(X7_3_4.java:22)
数组下标越界!
```

5．自定义两个异常类 NumberTooBigException 和 NumberTooSmallException，在其中定义各自的构造方法，分别打印输出“发生数字太大异常”和“发生数字太小异常”。然后在主类中定义一个带 throws 的方法 numberException(int x)，当 x>100 时通过 throw 抛出 NumberTooBigException 异常，当 x<0 时通过 throw 抛出 NumberTooSmallException 异常；最后在 main()方法中调用该方法，实现从键盘中输入一个整数，如果输入的是负数，引发 NumberTooSmallException 异常，如果输入的数大于 100，引发。NumberTooBigException 异常，否则输出“没有发生异常”。

【编程分析】本题主要考察自定义异常的方法。

第一步：定义异常类 NumberTooBigException

```
class NumberTooBigException extends Exception{
    NumberTooBigException(){
        super("发生数字太大异常  ");
    }
}
```

第二步：定义异常类 NumberTooSmallException

```
class NumberTooSmallException extends Exception{
    NumberTooSmallException(){
        super("发生数字太小异常");
    }
}
```

第三步：在主类 X7_3_5 中定义 numberException()方法。

```
public static void numberException(int x)
    throws NumberTooBigException, NumberTooSmallException{
        if(x>100)
            throw new NumberTooBigException();
        else if (x<0)
            throw new NumberTooSmallException();
        else
            System.out.println("没有异常发生")   ;
}
```

第四步：在 main()方法中调用 numberException()方法并捕捉和处理由此方法引起的异常。

【参考程序】

```
import java.io.*;
public class X7_3_5 {
```

```
        public static void main(String args[]) throws IOException{
            InputStreamReader isr = new InputStreamReader(System.in);
            BufferedReader br = new BufferedReader(isr);
            System.out.print("请输入一个整数：");
            int a = Integer.parseInt( br.readLine());
            try{
                numberException(a);
            }catch(NumberTooBigException e){
                e.printStackTrace();
            }catch(NumberTooSmallException e){
                e.printStackTrace();
            }
        }
        public static void numberException(int x) throws NumberTooBigException,
NumberTooSmallException{
                if(x>100)
                    throw new NumberTooBigException();
                else if (x<0)
                    throw new NumberTooSmallException();
                else
                    System.out.println("没有异常发生")   ;
        }
    }
    class NumberTooBigException extends Exception{
        NumberTooBigException(){
            super("发生数字太大异常  ");
        }
    }
    class NumberTooSmallException extends Exception{
        NumberTooSmallException(){
            super("发生数字太小异常");
        }
    }
```

【运行结果】

✧ 当输入的数值大于 100 时，运行结果为：

```
请输入一个整数：120
NumberTooBigException: 发生数字太大异常
        at X7_3_5.numberException(X7_3_5.java:18)
        at X7_3_5.main(X7_3_5.java:9)
```

✧ 当输入的数值小于 0 时，运行结果为：

```
请输入一个整数：-10
NumberTooSmallException: 发生数字太小异常
        at X7_3_5.numberException(X7_3_5.java:20)
        at X7_3_5.main(X7_3_5.java:9)
```

✧ 当输入的数值在 0～100 之间时，运行结果为：

请输入一个整数：25

没有异常发生

第 8 章

一、选择题

1．下列数据流中，属于输入流的一项是（ B ）。

A．从内存流向硬盘的数据流　　B．从键盘流向内存的数据流

C．从键盘流向显示器的数据流　　D．从网络流向显示器的数据流

【解析】输入流是指从外围设备流向主机（包括CPU和内存）的数据流。

2．Java语言提供处理不同类型流的类所在的包是（ D ）。

A．java.sql　　B．java.util　　C．java.net　　D．java.io

【解析】java.io 包是Java语言提供处理不同类型流的类所在的包。

3．不属于java.io包中的接口的是（ C ）。

A．DataInput　　B．DataOutput　　C．DataInputStream　　D．ObjectInput

【解析】DataInputStream是java.io包中的一个类，其他三个则是java.io包中的接口。

4．下列程序从标准输入设备读入一个字符，然后再输出到显示器，选择正确的一项填入“//x”处，完成要求的功能（ B ）。

```
import java.io.*;
    public class X8_1_4 {
        public static void main(String[] args) {
        char ch;
        try{
            //x
            System.out.println(ch);
        }
        catch(IOException e){
            e.printStackTrace();
        }
    }
}
```

A．ch = System.in.read();　　B．ch = (char)System.in.read();

C．ch = (char)System.in.readln();　　D．ch = (int)System.in.read();

【解析】System.in.read()方法返回的是字符对应的Unicode码，即返回的类型是int型，而ch是char类型，因此必须把方法的返回值强制转换为char类型才能把它赋值给ch变量。另外，System.in对象中没有readln()方法。

5．下列程序实现了在当前包dir815下新建一个目录subDir815，选择正确的一项填入程序的横线处，使程序符合要求（ D ）。

```
package dir815;
import java.io.*;
public class X8_1_5 {
    public static void main(String[] args){
        char ch;
        try{
            File path = _____;
            if(path.mkdir())
                System.out.println("successful!");
```

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

A．new File("subDir815");　　B．new File("dir815.subDir815");

C．new File("dir815\subDir815");　　D．new File("dir815/subDir815");

【解析】在程序中，目录之间的连接符是“\\”或“/”，因此首先排除B和C；而默认情况下，创建相对目录是在当前目录下进行，而dir815也在当前目录下，因此要在dir815下创建新的目录，就必须使用D的形式。

6．下列流中哪一个使用了缓冲区技术（　A　）？

A．BufferedOutputStream　　B．FileInputStream

C．DataOutputStream　　D．FileReader

【解析】只有BufferedOutputStream使用了缓冲区技术。

7．能读入字节数据进行Java基本数据类型判断过虑的类是（ C　　）。

A．BufferedInputStream　　B．FileInputStream

C．DataInputStream　　D．FileReader

【解析】DataInputStream类在读入字节数据时，进行Java基本数据类型判断过虑。

8．使用哪一个类可以实现在文件的任一个位置读写一个记录（ B　　）？

A．BufferedInputStream　　B．RandomAccessFile

C．FileWriter　　D．FileReader

【解析】只有RandomAccessFile才能实现在文件的任一个位置读写一个记录。

9．在通常情况下，下列哪个类的对象可以作为BufferedReader类构造方法的参数（　C　）？

A．PrintStream　　B．FileInputStream

C．InputStreamReader　　D．FileReader

【解析】InputStreamReader类的对象可以作为BufferedReader类构造方法的参数。

10．若文件是RandomAccessFile的实例f，并且其基本文件长度大于0，则下面的语句实现的功能是（　B　）。

```
f.seek(f.length()-1);
```

A．将文件指针指向文件的第一个字符后面

B．将文件指针指向文件的最后一个字符前面

C．将文件指针指向文件的最后一个字符后面

D．会导致seek()方法抛出一个IOException异常

【解析】通过调用f对象的length()方法，可以将文件指针指向文件的末尾，因此f.length()-1即指向文件的最后一个字符前面。

11．下列关于流类和File类的说法中错误的一项是（ B　　）。

A．File类可以重命名文件　　B．File类可以修改文件内容

C．流类可以修改文件内容　　D．流类不可以新建目录

【解析】只有流类可以修改文件内容，而File类则不能。

12．若要删除一个文件，应该使用下列哪个类的实例（ B　　）？

A．RandomAccessFile　　B．File

C．FileOutputStream　　D．FileReader

【解析】要删除文件以及查看文件属性等，应使用File类对象来实现。

13．下列哪一个是Java系统的标准输入流对象（　　）？

A．System.out　　B．System.in　　C．System.exit　　D．System.err

【解析】System.in是Java系统的标准输入流对象，而System.out和System.err则是Java系统的标准输出流和标准错误对象，System.exit则是System类的退出方法。

14．Java系统标准输出对象System.out使用的输出流是（ A ）。

A．PrintStream　　B．PrintWriter

C．DataOutputStream　　D．FileReader

【解析】System.out属性是PrintStream类型的对象。

二、填空题

1．Java的输入输出流包括 字节流 、 字符流 、 文件流 、 对象流 以及多线程之间通信的 管道流 。

2．凡是从外部设备流向中央处理器的数据流，称之为 输入 流；反之，称之为 输出 流。

3．java.io包中的接口中，处理字节流的有 DataInput 接口和 DataOutput 接口。

4．所有的字节输入流都从 InputStream 类继承，所有的字节输出流都从 OutputSteam 类继承。

5．与用于读写字节流的InputStream类和OutputStream类相对应，Java还提供了用于读写Unicode字符的字符流 Reader 类和 Writer 类。

6．对一般的计算机系统，标准输入通常是 键盘 ，标准输出通常是 显示器 。

7．Java系统事先定义好两个流对象，分别与系统的标准输入和标准输出相联系，它们是 System.in 和 System.out 。

8．System类的所有属性和方法都是 Static类型 的，即调用时需要以类名System为前缀。

9．Java的标准输入System.in是 InputStream 类的对象，当程序中需要从键盘读入数据的时候，只需调用System.in的 read 方法即可。

10．执行System.in.read()方法将从键盘缓冲区读入一个 字节 的数据，然而返回的却是16比特的 整形量 ，需要注意的是只有这个 整形量 的低位字节是真正输入的数据，其高位字节 全部为0 。

11．System.in只能从键盘读取 二进制 的数据，而不能把这些比特信息转换为整数、字符、浮点数或字符串等复杂数据类型的量。

12．Java的标准输出System.out是 PrintStream 类的对象。 PrintStream 类是过滤输出流类FilterOutputStream的一个子类，其中定义了向屏幕输送不同类型数据的方法 print() 和 println() 。

13．在Java中，标准错误设备用 System.err 表示。它属于PrintStream 类对象。

14．在计算机系统中，需要长期保留的数据是以 文件 的形式存放在磁盘、磁带等外存储设备中的。

15． 目录 是管理文件的特殊机制，同类文件保存在同一 目录 下可以简化文件的管理，提高工作效率。

16．Java语言的java.io包中的 File 类是专门用来管理磁盘文件和目录的。调用 File 类的方法则可以完成对文件或目录的常用管理操作，如创建文件或目录、删除文件或目录、查看文件的有关信息等。

17．File类也虽然在java.io包中，但它不是InputStream或者OutputStream的子类，因为它不负责 数据的输入输出 ，而专门用来管理 文件和目录 。

18．如果希望从磁盘文件读取数据，或者将数据写入文件，还需要使用文件输入输出流类 FileInputStream 和 FileOutputStream 。

19．Java系统提供的FileInputStream类是用于读取文件中的 字节 数据的 字节 文件输入流类；FileOutputStream类是用于向文件写入 字节 数据的 字节 文件输出流。

20．利用 DataInputStream 类和 DataOutputStream 类提供的成员方法可以方便地从文

件中读写不同类型的数据。

21．Java中的 RandomAccessFile 类提供了随机访问文件的功能，它继承了 Object 类，用 DataInput 和 DataOutput 接口来实现。

三、编程题

1．利用DataInputStream类和BufferedInputStream类编写一个程序，实现从键盘读入一个字符串，在显示器上显示前两个字符的Unicode码以及后面的所有字符。

【编程分析】本程序主要考察流类DataInputStream 和BufferedInputStream的使用方法。

第一步：创建字节输入流对象。

DataInputStream dis = new DataInputStream(System.in);

BufferedInputStream bis = new BufferedInputStream(dis);

第二步：利用字节输入流对象分三次读取数据，第一次读取一个字节，第二次读取一个字节，第三次将剩余字节全部读入字节数组b中，并将该数组转换为字符串显示出来。

注意：字节数组中元素的个数比实际输入元素的个数多两个，原因是数组最后都要添加回车和换行两个转义字符的Unicode码。

【参考程序】

```
import java.io.*;
public class X8_3_1 {
    public static void main(String[] args)throws IOException{
        DataInputStream dis = new DataInputStream(System.in);
        BufferedInputStream bis = new BufferedInputStream(dis);
        int code,count;
        byte[] b = new byte[256];
        code=bis.read();
        System.out.println("第一个字符的Unicode码为：" + code);
        code=bis.read();
        System.out.println("第二个字符的Unicode码为：" + code);
        count = bis.read(b);
        System.out.println("数组b中元素的值为：  ");
        for(int i=0; i<count; i++){
            System.out.print(b[i]+"\t");// 最后两个字符Unicode码是回车和换行
        }
        System.out.println("\n数组b中字符的个数为：" + count);
        String str = new String(b,0,count);
        System.out.println("剩余字符为：" + str);
    }
}
```

【运行结果】

```
abcdefgh
第一个字符的Unicode码为：97
第二个字符的Unicode码为：98
数组b中元素的值为：
99      100     101     102     103     104     13      10
数组b中字符的个数为：8
剩余字符为：cdefgh
```

2．编写一个程序，其功能是将两个文件的内容合并到一个文件中。

【编程分析】本题主要考察对文件流类FileReader和FileWriter的使用方法，实现从文件中读取数据，以及向文件中输入数据。

第一步：采用面向字符的文件流读出文件内容，使用FileReader类的read()方法，写文件内容使用FileWriter类的write()方法。

第二步：通过键盘方式输入要合并的两个源文件的文件名以及合并后的新文件名。

第三步：将两个源文件内容分别读出并写入到目标文件中。

【参考程序】

```
import java.io.*;
public class X8_3_2 {
    public static void main(String args[]) {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        File fin1, fin2, fout;
        try{
            System.out.print("Input the first input file name: ");
            fin1 = new File(br.readLine());
            if(!fin1.exists()){
                System.out.println("The file doesn't exist! ");
                System.exit(0);
            }
            System.out.print("Input the second input file name: ");
            fin2 = new File(br.readLine());
            if(!fin2.exists()){
                System.out.println("The file doesn't exist! ");
                System.exit(0);
            }
            System.out.print("Input the output file name: ");
            fout = new File(br.readLine());
            if((new JoinFiles(fin1,fin2,fout)) != null){    // 合并文件
                System.out.println("Successful in joining files!");
            }
            else{
                System.out.println("fail in joining files!");
            }
        }catch(IOException e){
            e.printStackTrace();
        }
    }
}
class JoinFiles{
    JoinFiles(File fin1, File fin2, File fout) throws IOException{
        FileReader fr1 = new FileReader(fin1);
        FileReader fr2 = new FileReader(fin2);
```

```
            FileWriter fw = new FileWriter(fout);
            int ch;
            while((ch=fr1.read()) != -1)
                fw.write(ch);
            while((ch=fr2.read()) != -1)
                fw.write(ch);
            fr1.close();
            fr2.close();
            fw.close();
        }
    }
```

【运行结果】

```
Input the first input file name: f1.txt
Input the second input file name: f2.txt
Input the output file name: fout.txt
Successful in joining files!
```

3．编写一个程序实现以下功能：

（1）产生5000个1～9999之间的随机整数，将其存入文本文件a.txt中。

（2）从文件中读取这5000个整数，并计算其最大值、最小值和平均值并输出结果。

【编程分析】本题主要考察利用 FileOutputStream、DataOutputStream、FileInputStream、DataInputStream 等类实现对文件的操作。

第一步：产生5000个1～9999之间的随机整数，将其存入文本文件a.txt中，本参考程序利用方法“genRandom(File f)”来实现，本方法使用了FileOutputStream和DataOutputStream两个类。

第二步：将文件中的数据取出进行计算最大值、最小值、平均值以及求和，本参考程序利用方法“calculate(File f)”来实现，本方法使用了FileInputStream和DataInputStream两个类。

【参考程序】

```
import java.io.*;
public class X8_3_3 {
    static int max, min, sum = 0;
    static int[] a = new int[5000];
    public static void main(String args[]) {
        File f = new File("a.txt");
        if(f == null){
            System.out.println("Can't create the file");
            System.exit(0);
        }
        genRandom(f);
        calculate(f);
    }
    static void genRandom(File f){     // 产生随机数方法
        try{
            FileOutputStream fos = new FileOutputStream(f);
            DataOutputStream dos = new DataOutputStream(fos);
```

```
                for(int i=0; i<5000; i++){
                    dos.writeInt((int)(Math.random()*10000));
                }
                dos.close();
            }catch(FileNotFoundException e){
                e.printStackTrace();
            }catch(Exception e){
                e.printStackTrace();
            }
        }
        static void calculate(File f){  // 计算最大值、最小值、平均值以及求和方法
            try{
                FileInputStream fis = new FileInputStream(f);
                DataInputStream dis = new DataInputStream(fis);
                int i;
                for(i=0; i<5000; i++){
                    a[i] = dis.readInt();
                }
                dis.close();
                max = a[0];
                min = a[0];
                for(i=0; i<5000; i++){
                    if(max <a[i]) max = a[i];
                    if(min>a[i]) min = a[i];
                    sum += a[i];
                }
            }catch(FileNotFoundException e){
                e.printStackTrace();
            }catch(Exception e){
                e.printStackTrace();
            }
            int average = sum/5000;
            System.out.println("max = "+max+"\tmin="+min);
            System.out.println("sum = "+sum+"\taverage="+average);
        }
    }
```

【运行结果】

```
max = 9997          min=6
sum = 25031340    average=5006
```

4．编写一个程序，将Fibonacii数列的前20项写入一个随机访问文件，然后从该文件中读出第2、4、6等偶数位置上的项并将它们依次写入另一个文件。

【编程分析】本程序主要考察RandomAccessFile文件流类的使用方法。

第一步：创建RandomAccessFile文件流类对象raf，让它指向文件“fout.txt”，并向该文件中写入Fibonacii数列的前20项。

第二步：读取“fout.txt”文件中第2、4、6等偶数位置上的项，并将它们存入数组fib2中。

第三步：让文件流类对象raf指向文件“fin.txt”，并将数组fib2中的数据写入其中。

【参考程序】

```
import java.io.*;
public class X8_3_4 {
    public static void main(String args[]) {
        final int M = 20;
        int[] fib = new int[M];
        int[] fib2 = new int[M];
        long fp;
        fib[0] = 1;
        fib[1] = 1;
        int i;
        for(i=2; i<M; i++)
            fib[i] = fib[i-1]+fib[i-2];
        try{
            RandomAccessFile raf = new RandomAccessFile("fout.txt","rw");
            System.out.println("fout.txt中的内容为：");
            for(i=0; i<M; i++){
                raf.writeInt(fib[i]);
                System.out.print(fib[i]+"\t");
            }
            for(i=1; i<M; i+=2){
                fp = i*4;
                raf.seek(fp);
                fib2[i/2]=raf.readInt();
            }
            raf.close();
            raf = new RandomAccessFile("fin.txt","rw");
            System.out.println("fin.txt中的内容为：");
            for(i=0; i<M/2; i++){
                System.out.print(fib2[i]+"\t");
                raf.writeInt(fib2[i]);
            }

            raf.close();
        }catch(FileNotFoundException e){
            e.printStackTrace();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

【运行结果】

fout.txt中的内容为：

1 1 2 3 5 8 13 21 34 55
89 144 233 377 610 987 1597 2584 4181
6765

fin.txt中的内容为：

1 3 8 21 55 144 377 987 2584
6765

第 9 章

一、选择题

1．下列说法中，正确的一项是（ A ）。

A．单处理机的计算机上，2 个线程实际上不能并发执行

B．单处理机的计算机上，2 个线程实际上能够并发执行

C．一个线程可以包含多个进程

D．一个进程只能包含一个线程

【解析】单处理机的计算机上通过一个极短的固定时间段或者在线程等待时，切换到另一个线程，这种调度过程时间极短，看上去像是并发执行。

2．下列说法中，错误的一项是（ A ）。

A．线程就是程序　　B．线程是一个程序的单个执行流

C．多线程是指一个程序的多个执行流　　D．多线程用于实现并发

【解析】线程是一个程序的单个执行流，而不是程序本身。而多线程作为实现并发的一个重要手段，是一个程序的多个执行流。

3．下列关于 Thread 类的线程控制方法的说法中错误的一项是（ C ）。

A．线程可以通过调用 sleep()方法使比当前线程优先级低的线程运行

B．线程可以通过调用 yield()方法使和当前线程优先级一样的线程运行

C．线程的 sleep()方法调用结束后，该线程进入运行状态

D．若没有相同优先级的线程处于可运行状态，线程调用 yield()方法时，当前线程将继续执行

【解析】线程的 sleep()方法调用结束后，该线程进入就绪状态，而不是运行状态。

4．方法 resume()负责恢复下列哪一个线程的执行（ D ）？

A．通过调用 stop()方法而停止的线程　　B．通过调用 sleep()方法而停止的线程

C．通过调用 wait()方法而停止的线程　　D．通过调用 suspend()方法而停止的线程

【解析】通过调用 suspend()方法而停止的线程需要调用 resume()恢复。

5．下面的哪一个关键字通常用来对对象加锁，从而使得对对象的访问是排他的（ C ）？

A．serialize　　B．transient　　C．synchronized　　D．static

【解析】synchronized 用来对对象加锁，从而使得对对象的访问是排他的。

6．下列说法中，错误的一项是（ A ）。

A．线程一旦创建，则立即自动执行

B．线程创建后需要调用 start()方法，将线程置于可运行状态

C．调用线程的 start()方法后，线程也不一定立即执行

D．线程处于可运行状态，意味着它可以被调度

【解析】线程创建后需要调用 start()方法，将线程置于可运行状态。

7．下列说法中，错误的一项是（ A ）。

A．Thread 类中没有定义 run()方法　　B．可以通过继承 Thread 类来创建线程
C．Runnable 接口中定义了 run()方法　　D．可以通过实现 Runnable 接口创建线程

【解析】Thread 类和 Runnable 接口中都定义了 run()方法，而 start()方法只有 Thread 类中进行了定义，而 Runnable 接口中没有定义。

8．Thread 类定义在下列哪个包中（　B　）？

A．java.io　　B．java.lang　　C．java.util　　D．java.awt

【解析】Thread 类定义在 java.lang 包中，因此使用时可以不用显式加载。

9．Thread 类的常量 NORM_PRIORITY 代表的优先级是（　C　）。

A．最低优先级　　B．最高优先级　　C．普通优先级　　D．不是优先级

【解析】NORM_PRIORITY 代表的优先级是普通优先级。

10．下列关于线程优先级的说法中，错误的一项是（　D　）。

A．MIN_PRIORITY 代表最低优先级　　B．MAX_PRIORITY 代表最高优先级
C．NORM_PRIORITY 代表普通优先级　　D．代表优先级的常数值越大优先级越低

【解析】代表优先级的常数值越大优先级越高

二、填空题

1．多线程是指程序中同时存在着 多 个执行体，它们按几条不同的执行路线共同工作，独立完成各自的功能而互不干扰。

2．每个 Java 程序都有一个缺省的主线程，对于 Application 类型的程序来说，主线程是方法 main() 执行的线程。

3．Java 语言使用 Thread 类及其子类的对象来表示线程，新建的线程在它的一个完整的生命周期中通常要经历 新生 、 就绪 、 运行 、 阻塞 和 死亡 等五种状态。

4．在 Java 中，创建线程的方法有两种：一种方法是通过创建 Thread 类的子类来实现，另一种方法是通过实现 Runnable 接口的类来实现。

5．用户可以通过调用 Thread 类的方法 setPriority() 来修改系统自动设定的线程优先级，使之符合程序的特定需要。

6．start() 方法将启动线程对象，使之从新建状态转入就绪状态并进入就绪队列排队。

7．Thread 类和 Runnable 接口中共有的方法是 run() ，只有 Thread 类中有而 Runnable 接口中没有的方法是 start() ，因此通过实现 Runnable 接口创建的线程类要想启动线程，必须在程序中创建 Thread 类的对象。

8．在 Java 中，实现同步操作的方法是在共享内存变量的方法前加 synchronized 修饰符。

9．线程的优先级是一个在 1 到 10 之间的正整数，数值越大，优先级越 高 ，未设定优先级的线程其优先级取缺省值 5 。

10．Thread 类中代表最高优先级的常量是 MAX_PRIORITY ，表示最低优先级的常量是 MIN_PRIORITY 。

三、编程题

1．编写一个有两个线程的程序，第一个线程用来计算 2～100000 之间的素数的个数，第二个线程用来计算 100000～200000 之间的素数的个数，最后输出结果。

【编程分析】本程序主要考察如何实现多线程编程。

第一步：创建线程类 CalPrime，在该类中实现素数个数的计算。

第二步：在主类的 main()方法中创建线程类对象并启动线程。

【参考程序】

```
import java.io.*;
public class X9_3_1 {
    public static void main(String[] args)throws IOException{
```

```
            CalPrime thr1 = new CalPrime(2,100000);
            CalPrime thr2 = new CalPrime(10000,200000);
            thr1.start();
            thr2.start();
        }
    }
    class CalPrime extends Thread{
        int from, to, count=0;
        public CalPrime(int from, int to){  //  构造方法
            this.from = from;
            this.to = to;
        }
        public void run(){   //  该方法主要计算从 from 到 to 之间素数的个数
            for(int i=from; i<to; i++){
                int m = (int)Math.sqrt(i);
                boolean isPrime = true;
                for(int j=2; j<=m; j++){
                    if(i%j==0){
                        isPrime = false;
                        break;
                    }
                }
                if(isPrime) count++;
            }
            System.out.println(from + "~"+to+"之间的素数个数为："+count);
        }
    }
```

【运行结果】

2~100000 之间的素数个数为：9592

100001~200000 之间的素数个数为：8392

2．编写一个龟兔赛跑的多线程 Applet 程序。

【编程分析】本程序主要考察在 Applet 中实现多线程的问题。

第一步：创建一个继承 Applet 类、实现 Runnable 接口的类，该类属性包括：Thread 类型的 rabbit 和 tortoise，String 类型的 rab 和 tot，int 型的 xr、yr、 xt 和 yt。该类方法包括：init()、start()、paint()以及 run()。

第二步：在 init()方法中，初始化属性 xr、yr、 xt 和 yt。

第三步：在 start()方法中创建线程 rabbit 和 tortoise，并启动这两个线程。

第四步：在 paint()方法中，绘制“兔子”和“乌龟”两个字符串。

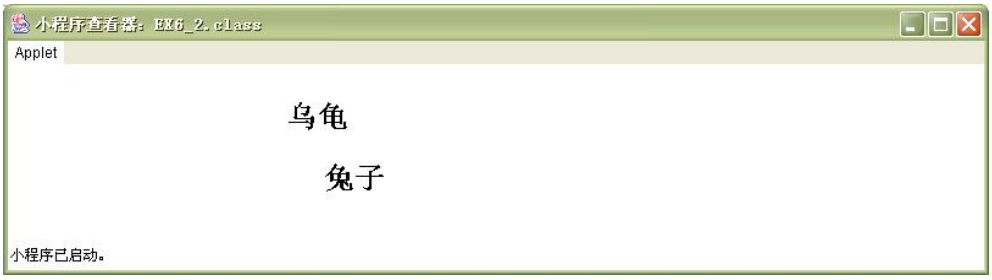
第五步：在 run()方法中，使“兔子”和“乌龟”两个字符串的 x 坐标 xt 和 xr 的值不断发生变化，并调用 repaint()方法不断刷新 Applet 界面，使字符串的位置不断发生变化，看起来就像两个字符串赛跑一样。

【参考程序】

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

```
public class X9_3_2 extends Applet implements Runnable{
  Thread rabbit, tortoise;
  Font font = new Font("宋体",Font.BOLD, 24);
  String rab = "兔子";
  String tot = "乌龟";
  int xr, yr, xt, yt;
  public void init() {
        xr=this.getSize().width;
        yr = this.getSize().height;
        xt = this.getSize().width;
        yt = this.getSize().height/2;
        this.setSize(800,200);
  }
  public void start(){
        if(rabbit == null){
              rabbit = new Thread(this);
              rabbit.start();
        }
        if(tortoise == null){
              tortoise = new Thread(this);
              tortoise.start();
        }
  }
  public void paint(Graphics g) {
        g.setFont(font);
        g.drawString(rab,xr,yr);
        g.drawString(tot,xt,yt);
  }
  public void run(){
        for(; ;){
              xr -= 10;
              if(xr == 0) xr = this.getSize().width;
              xt -= 5;
              if(xt == 0) xt = this.getSize().width;
              repaint();
              try{
                    rabbit.sleep(100);
                    tortoise.sleep(50);
              }catch(Exception e){}
        }
  }
}
```

【运行结果】



3．编写一个程序，让一个小球在 Applet 中滚动，当碰到边缘时，则选择一个角度反弹回去。

第 10 章

一、选择题

1．下列说法中错误的一项是（　B　）。

A．构件是一个可视化的能与用户在屏幕上交互的对象

B．构件能够独立显示出来

C．构件必须放在某个容器中才能正确显示

D．一个按钮可以是一个构件

【解析】构件不能独立显示，它必须放在某个容器中才能正确显示。

2．进行 Java 基本 GUI 设计需要用到的包是（　C　）。

A．java.io　　B．java.sql　　C．java.awt　　D．java.rmi

【解析】进行 Java 基本 GUI 设计需要用到的包是 java.awt 和 javax.swing

3．Container 是下列哪一个类的子类（　D　）？

A．Graphics　　B．Window　　C．Applet　　D．Component

【解析】Container 类是由 Component 类派生的。

4．java.awt.Frame 的父类是（　B　）。

A．java.util.Window　　B．java.awt Window

C．java.awt Panel　　D．java.awt.ScrollPane

【解析】java.awt.Frame 的父类 java.awt Window。

5．下列哪个方法可以将 MenuBar 加入 Frame 中（　D　）？

A．setMenu()　　B．addMenuBar()　　C．add()　　D．setMenuBar()

【解析】可以将 MenuBar 加入 Frame 中的方法是 setMenuBar()。

6．下列叙述中，错误的一项是（　D　）。

A．采用 GridLayout 布局，容器中的每个构件平均分配容器空间

B．采用 GridLayout 布局，容器中的每个构件形成一个网络状的布局

C．采用 GridLayout 布局，容器中的构件按照从左到右、从上到下的顺序排列

D．采用 GridLayout 布局，容器大小改变时，每个构件不再平均分配容器空间

【解析】采用 GridLayout 布局，容器大小改变时，每个构件平均分配容器空间。

7．当单击鼠标或拖动鼠标时，触发的事件是（　D　）。

A．KeyEvent　　B．ActionEvent　　C．ItemEvent　　D．MouseEvent

【解析】对鼠标操作，触发的事件是 MouseEvent 事件。

8．下列哪一项不属于 Swing 的顶层组件（　C　）？

A．JApplet　　B．JDialog　　C．JTree　　D．Jframe

【解析】JTree　　只有在容器中才能显示，它不属于 swing 的顶层组件。

9．下列说法中错误的一项是（ D ）。

A．在实际编程中，一般使用的是 Component 类的子类

B．在实际编程中，一般使用的是 Container 类的子类

C．Container 类是 Component 类的子类

D．容器中可以放置构件，但是不能够放置容器

【解析】容器中既可以放置构件，也可以放置容器。

10．下列哪一项不属于 AWT 布局管理器（ D ）？

A．GridLayout B．CardLayout C．BorderLayout D．BoxLayout

【解析】BoxLayout 属于 swing 布局管理器，不属于 AWT 布局管理器。

11．下列说法中错误的一项是（ A ）。

A．MouseAdapter 是鼠标运动适配器 B．WindowAdapter 是窗口适配器

C．ContainerAdapter 是容器适配器 D．KeyAdapter 是键盘适配器

【解析】MouseAdapter 是鼠标适配器，而 MouseMotionAdapte 才是鼠标运动适配器。

12．布局管理器可以管理构件的哪个属性（ A ）？

A．大小 B．颜色 C．名称 D．字体

【解析】布局管理器可以管理构件的位置和大小，而不能管理构件的其他属性。

13．编写 AWT 图形用户界面的时候，一定要 import 的语句是（ B ）。

A．import java.awt; B．import java.awt.*;

C．import javax.awt D．import javax.swing.*;

【解析】“import java.awt.*;”语句的含义是加载 awt 包中的所有类，而其他都不是。

14．在类中若要处理 ActionEvent 事件，则该类需要实现的接口是（ B ）。

A．Runnable B．ActionListener C．Serializable D．Event

【解析】处理 ActionEvent 事件的类需要实现的接口是 ActionListener，它其中包含了 actionPerformed()方法。

15．下列不属于 java.awt 包中的基本概念的一项是（ C ）。

A．容器 B．构件 C．线程 D．布局管理器

【解析】线程不属于 java.awt 包中的基本概念的一项，其他三个都是。

16．下列关于 AWT 构件的说法中错误的一项是（ D ）。

A．Frame 是顶级窗口，它无法直接监听键盘输入事件

B．对话框需要依赖于其他窗口而存在

C．菜单只能被添加到菜单栏中

D．可以将菜单添加到任意容器的某处

【解析】菜单只能添加到 Applet、Frame 等容器中，不能添加到任意容器的某处。

17．JPanel 的默认布局管理器是（ C ）。

A．BorderLayout B．GridLayout C．FlowLayout D．CardLayout

【解析】Panel、JPanel 和 Applet 的默认布局管理器都是 FlowLayout。

18．下列说法中错误的是（ B ）。

A．在 Windows 系统下，Frame 窗口是有标题、边框的

B．Frame 的对象实例化后，没有大小，但是可以看到

C．通过调用 Frame 的 setSize()方法来设定窗口的大小

D．通过调用 Frame 的 setVisible(true)方法来设置窗口为可见

【解析】Frame 的对象实例化后，没有大小，也不能看到，只有通过调用 Frame 的 setSize()和 setVisible(true)方法才能设定窗口的大小和可见性。

19．下列说法中错误的是（ D ）。

A．同一个对象可以监听一个事件源上多个不同的事件

B．一个类可以实现多个监听器接口

C．一个类中可以同时出现事件源和事件处理者

D．一个类只能实现一个监听器接口

【解析】一个类可以实现多个监听器接口，从而实现对多个事件的监听。

20．下列选项中不属于容器的一项是（　　）。

A．Window　　B．Panel　　C．FlowLayout　　D．ScrollPane

【解析】FlowLayout 类属于布局管理器，而不属于容器。

二、填空题

1. Java 编程语言是一种跨平台的编程语言，在编写图形用户界面方面，也要支持 跨平台 功能。

2．Java 的图形用户界面技术经历了两个发展阶段，分别通过提供 awt 开发包和 swing 开发包来体现。

3．在进行界面设计的时候，只要掌握好 AWT 和 Swing 的三点思路，就能编写出较好的图形用户界面：首先是 界面中的构件如何放置 ，其次是 如何让构件响应用户的操作 ，第三是 掌握每种构件的显式效果和响应用户操作 。

4．java.awt 包提供了基本的 java 程序的 GUI 设计工具，主要包括下述三个概念，它们分别是： 构件 、 容器 和 布局管理器 。

5．构件不能独立地显示出来，必须将构件放在一定的 容器 中才可以显示出来。

6．容器本身也是一个 构件 ，具有构件的所有性质，另外还具有放置其他 构件 和 容器 的功能。

7．容器中的布局管理器负责各个构件的 位置 和 大小 ，因此用户无法在这种情况下设置构件的这些属性。

8．如果用户确实需要亲自设置构件大小或位置，则应取消该容器的布局管理器，方法为 setLayout(null) 。

9．所有的构件都可以通过 add() 方法向容器中添加构件。

10．有 3 种类型的容器： Window 、 Panel 、 ScrollPane 。

11．FlowLayout 类是 java.lang.Object 直接子类。其布局策略是：将容器中的构件按照加入的先后顺序从 左 向 右 排列，当一行排满之后就转到下一行继续从 左 向 右 排列，每一行中的构件都 居中 排列。它是 Panel 和 Applet 缺省使用的布局编辑策略。

12．对于一个原本不使用 FlowLayout 布局编辑器的容器，若需要将其布局策略改为 FlowLayout，可以使用 setLayout(new FlowLayout()) 方法。

13．BorderLayout 类的布局策略是：把容器内的空间划分为 东 、 西 、 南 、 北 、 中 五个区域，它们分别用字符串常量 East 、 West 、 South 、 North 、 Center 表示。

14．BorderLayout 是 Window 、 Frame 、 Dialog 和 JApplet 的缺省布局策略。

15．在事件处理的过程中，主要涉及 3 类对象： 事件 、 事件源 和 事件处理者 。

16．事件类主要有两个： java.util.EventObject 类以及 java.awt.AWTEvent 类。

17．根据监听器器和注册监听器所在的类之间的关系，我们可以把事件处理分为以下几种情况：利用 外部类 对象、 本类 对象、 内部类 对象和 匿名内部类 对象处理事件。

18．标准构件是由 容器 和 基本构件 构成，容器是能够容纳其他构件的对象，而基本构件是放置在容器中而不能在其内部存放其他构件的对象。

19．按钮可以引发 ActionEvent 事件，TextField 可产生 TextEvent 和 ActionEvent 事件，下拉列表可产生 ItemEvent 项目事件。当用户单击复选框使其选中状态发生变化时就

会引发 ItemEvent 类代表的选择事件。滚动条可以引发 AjustmentEvent 类代表的调整事件。

20. ActionEvent 事件类包含 ACTION_PERFOMED 事件，该事件通过 ActionListener 接口进行监听，通过调用 addActionListener() 方法将事件源注册到监听器，通过调用 actionPerformed(ActionEvent e) 方法实现监听后的动作，通过调用 getSource()方法可以获得发生事件的事件源对象，调用 getActionCommand() 方法可以获取引发事件动作的命令名。

21. 通常在 itemStateChanged(ItemEvent e)方法里，会调用 e.getItemSelectable() 方法获得产生这个选择事件的列表（List）对象的引用，再利用列表对象的方法 getSelectedIndex() 或 getSelectedItem() 就可以方便地得知用户选择了列表的哪个选项。

22. 列表的双击事件 不能 （能/不能）覆盖单击事件。当用户双击一个列表选项时，首先产生一个 ItemEvent 事件，然后再产生一个 ActionEvent 事件。

22. 调整事件（AdjustmentEvent）类只包含一个事件—— AJUSTMENT_VALUE_CHANGED 事件， AjustmentEvent.TRACK 代表鼠标拖动滚动条滑块的动作。

23. 调用 MouseEvent 对象的 getID() 方法就可以知道用户引发的是哪个具体的鼠标事件。

24. 在菜单项之间增加一条横向分隔线的方法是 addSeparator() 。

25. 将菜单项添加到菜单中以及将菜单添加的菜单栏中所用的方法都是 add() ，将菜单栏添加到窗口中的方法是 setMenuBar() 。

26. 对话框构件一般可以接受 ComponentEvent 事件和 FocusEvent 事件。

27. 创建字体后，可以用 Graphics 类的成员方法 setFont() 来设置自己希望使用的字体。

28. Java 中可以利用 Graphics2D 类的 drawImage() 方法显示图像。

29. 在 Swing 中完全可以使用 java.awt.event 包中的各种类进行事件处理，同时它也可以使用 javax.swing.event 包中的类处理事件，而 AWT 则只能使用 java.awt.event 包中的各种类进行事件处理。

30. 可将 JOptionPane 类的对话框分为 4 种类型，分别是只给出提示信息的 Message Dialog 、要求用户进行确认的 Confirm Dialog 、可输入数据的 Input Dialog 和由用户自己定义类型的 Option Dialog 。

三、编程题

1. 创建一个 Frame 类型窗口，在窗口中添加 2 个不同颜色的 Panel 面板，每个面板中添加 2 个按钮构件。

【编程分析】本程序主要考察窗口、面板以及按钮的创建及布局问题。

第一步：首先定义一个主类，让该类继承 Frame 类。

第二步：定义该类的数据成员，包括两个 Panel 对象，一个长度为 4 的 Button 对象数组。

第三步：创建类的工作方法，在方法中创建各个对象、设置对象属性、布局整个界面、设置窗口大小并显示界面。

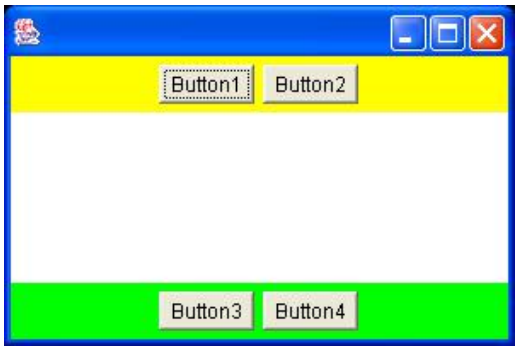
第四步：在类的 main()方法中创建本类对象，从而显示整个窗口界面。

【参考程序】

```
import java.io.*;
import java.awt.*;
public class X10_3_1 extends Frame{
        Panel pn1,pn2;       // 定义面板
        Button[] bt = new Button[4];     // 定义按钮数组
        public static void main(String[] args)throws IOException{
              new X10_3_1 ();
```

```
        }
        public X10_3_1 (){
            pn1 = new Panel();        // 创建面板对象
            pn2 = new Panel();
            pn1.setBackground(Color.yellow);       // 设置面板背景颜色
            pn2.setBackground(Color.green);
            for(int i=0; i<4; i++){    // 创建按钮对象
                bt[i] = new Button("Button"+(i+1));
            }
            pn1.add(bt[0]);      // 向面板中添加按钮，面板的默认布局为 FlowLayout
            pn1.add(bt[1]);
            pn2.add(bt[2]);
            pn2.add(bt[3]);
            add(pn1,BorderLayout.NORTH);
                                  // 向窗口添加面板，窗口默认布局为 BorderLayout
            add(pn2,BorderLayout.SOUTH);
            this.setSize(300,200);    // 设置窗口大小
            this.setVisible(true);      // 显示窗口
        }
    }
```

【运行结果】



2．创建一个 Frame 类型窗口，采用 GridLayout 布局，在窗口中创建一个计算器的界面。

【编程分析】本程序主要考察布局管理问题。

第一步：创建一个主类，其中定义两个面板，一个文本框，一个二维按钮数组，一个二维字符串数组。一个构造方法、一个 main()方法。

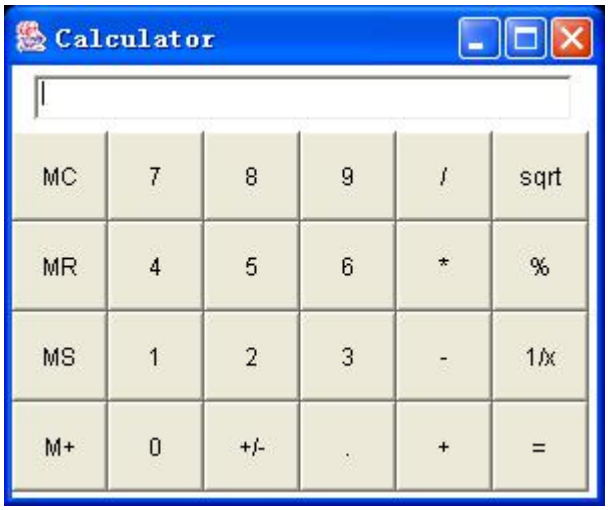
第二步：在构造方法中创建控件，实现布局。

【参考程序】

```
import java.io.*;
import java.awt.*;
public class X10_3_2 extends Frame{
    Panel pn1,pn2;       // 定义两个面板
    TextField tf;          // 定义文本框控件
    Button[][] bt = new Button[4][6];  // 定义按钮对象二维数组
    String[][] str = {{"MC","7","8","9","/","sqrt"},{"MR","4","5","6","*","%"},
```

```
                {"MS","1","2","3","-","1/x"},{"M+","0","+/-",".","+","="}};
    public static void main(String[] args)throws IOException{
        new X10_3_2();
    }
    public X10_3_2(){
        pn1 = new Panel();        // 创建面板
        pn2 = new Panel();
        tf = new TextField(35);  // 创建文本框
        pn1.add(tf);
        pn2.setLayout(new GridLayout(4,6));   // 设置 pn2 面板的布局
        for(int i=0; i<4; i++){    // 创建 24 个按钮控件并添加到 pn2 面板中
            for(int j=0; j<6; j++){
                bt[i][j] = new Button(str[i][j]);
                pn2.add(bt[i][j]);
            }
        }
        add(pn1,BorderLayout.NORTH);        // 将 pn1 面板添加到窗口北面
        add(pn2,BorderLayout.CENTER);       //将 pn2 面板添加到窗口中间
        this.setTitle("Calculator");     // 设置窗口标题
        this.setSize(300,250);           // 设置窗口大小
        this.setVisible(true);           // 显示窗口
    }
}
```

【运行结果】



3．创建两个 Frame 类型窗口，在第一个窗口中添加一个按钮，当单击按钮时打开第二个窗口，当单击两个窗口中的关闭按钮时能关闭窗口。

【编程分析】本程序主要考察窗口事件、按钮事件的实现方法。

第一步：定义一个能创建两个窗口的类 TwoFrames，由于该类中的两个窗口能够关闭，并且对点击按钮有响应，因此让该类继承 WindowAdapter 类，并实现 ActionListener 接口。在类中创建两个窗口，并在第一个窗口中添加一个按钮控件；重新定义继承自

WindowAdapter 类的方法 windowClosing()，实现关闭窗口的目的，实现 ActionListener 接口中的方法 actionPerformed()，从而实现点击按钮时打开第二个窗口。

第二步：在主类中的 main()方法中创建类 TwoFrames 的对象。

【参考程序】

```
import java.awt.*;
import java.awt.event.*;
public class X10_3_3 {   // 定义主类
      public static void main(String[] args) {
            new TwoFrames();
      }
}
class TwoFrames extends WindowAdapter implements ActionListener{ // 定义窗口类
  Frame fr1, fr2;       // 定义两个窗口
  Button bt;            // 定义按钮
  TwoFrames(){       // 构造方法
        fr1 = new Frame("First Frame");
        fr2 = new Frame("Second Frame");
        bt = new Button("Open Second Frame");
        bt.addActionListener(this);
        fr1.add(bt, BorderLayout.NORTH);
        fr1.addWindowListener(this );
        fr2.addWindowListener(this );
        fr1.setSize(200, 150);
        fr2.setSize(200,150);
        fr1.setVisible(true);
  }
  public void actionPerformed(ActionEvent e){     // 处理点击按钮事件
        if(e.getSource() == bt){
            fr2.setVisible(true);
        }
  }
  public void windowClosing(WindowEvent e){    // 处理关闭窗口操作
        System.exit(0);
  }
}
```

【运行结果】

4．编写一个能在窗口中同时响应鼠标事件和键盘事件的程序，能对鼠标的各种动作进行监听，对键盘的输入做出相应的反应。

【编程分析】本程序主要考察对鼠标、键盘事件的处理问题。

第一步：定义一个 MouseKeyFrame 类，该类主要有三个功能，（1）创建窗口。（2）处理鼠标事件。（3）处理键盘事件。创建窗口主要在类的构造方法中实现；处理鼠标事件主要在内部类 MouseHandler 中实现，该类实现了 MouseListener 和 MouseMotionListener 两个接口；处理键盘事件主要在内部类 KeyHandler 中实现，该类继承了 Canvas 类，实现了 KeyListener 接口。Canvas 类对象是接受键盘事件的容器，KeyListener 接口中包含了处理键盘事件的方法。

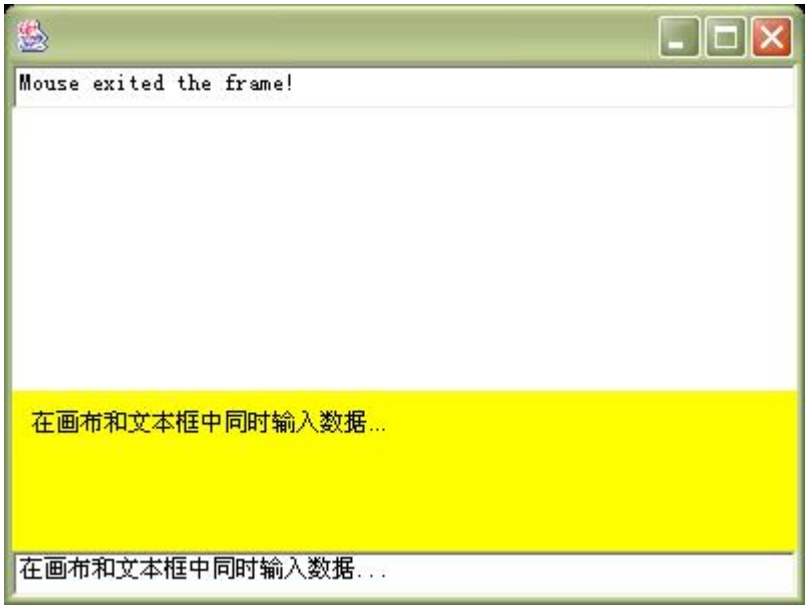
第二步：创建主类 X10_3_4，在该类的 main()方法中创建 MouseKeyFrame 类对象。

【参考程序】

```
import java.awt.*;
import java.awt.event.*;
public class X10_3_4 {       // 定义主类
      public static void main(String[] args) {
        new MouseKeyFrame();
      }
}
class MouseKeyFrame extends Frame {  // 定义创建窗口、处理鼠标和键盘事件的类
  Panel pn;
  TextField tf1, tf2;
  KeyHandler kh;
  MouseKeyFrame(){
        pn = new Panel(new BorderLayout() );
        tf1 = new TextField(30);
        tf2 = new TextField(30);
        kh = new KeyHandler(new Dimension(200,80), Color.yellow );
        pn.add(kh, BorderLayout.CENTER);
        pn.add(tf2, BorderLayout.SOUTH);
        this.add(tf1, BorderLayout.NORTH);
        this.add(pn, BorderLayout.SOUTH);
        this.addMouseListener(new MouseHandler() );
        this.addMouseMotionListener(new MouseHandler() );
        this.addWindowListener(new WindowAdapter(){// 关闭窗口事件处理
              public void windowClosing(WindowEvent e){
                    System.exit(0);
              }
        }       );
        this.setSize(400, 300);
        this.setVisible(true);
  }
  class MouseHandler implements MouseListener, MouseMotionListener{
                                                          // 处理鼠标事件内部类
```

```
        // 以下 5 个方法是实现 MouseListener 接口中的方法
        public void mouseEntered(MouseEvent e){
                tf1.setText("Mouse entered the frame!");
          }
          public void mouseExited(MouseEvent e){
                tf1.setText("Mouse exited the frame!");
          }
          public void mousePressed(MouseEvent e){
                tf1.setText("Mouse pressed in the frame!");
          }
          public void mouseReleased(MouseEvent e){
                tf1.setText("Mouse released in the frame!");
          }
          public void mouseClicked(MouseEvent e){
                tf1.setText("Mouse clicked in the frame!");
          }
          // 以下 2 个方法是实现 MouseMotionListener 接口中的方法
          public void mouseMoved(MouseEvent e){法
                tf1.setText("Mouse moved in the frame!");
          }
          public void mouseDragged(MouseEvent e){
                tf1.setText("Mouse dragged in the frame!");
          }
    }
    class KeyHandler extends Canvas implements KeyListener{// 处理键盘事件内部类
        StringBuffer sb ;
        KeyHandler(Dimension d, Color c){
            sb = new StringBuffer();
            this.setSize(d);
            this.setBackground(c);
            this.addKeyListener(this);
        }
        public void keyPressed(KeyEvent e){          }
        public void keyReleased(KeyEvent e){    }
        public void keyTyped(KeyEvent e){
            tf2.setText("" + sb.append(e.getKeyChar()) );
            repaint();
        }
        public void paint(Graphics g){
            g.drawString("" + sb, 10, 20);
        }
    }
}
```

【运行结果】当鼠标进入窗口（上面白色区域）时，将在上面的文本框中显示鼠标动作，当用鼠标点击画布（下面有色区域）使画布获得焦点后，即可以用键盘输入数据，输入的数据在画布和下面的文本框中同时显示。如下图：



5．编写一个测试计算是否正确的程序，窗口中包含 3 个按钮、3 个单行文本输入区、一个下拉列表框，当单击第 1 个按钮时在第 1 个单行文本输入区中产生一个随机数，当单击第 2 个按钮时在第 2 个单行文本输入区中产生一个随机数，在下拉列表框中选择一种运算符，如+、-、*、/等，然后单击第 3 个按钮，将计算结果显示在第 3 个单行文本输入区中。

【编程分析】本程序主要考察按钮、文本框、下拉列表框的应用以及随机数的产生和对 ActionEvent 时间的处理方法。

第一步：创建一个 RandomFrame 类，该类继承 Frame 类，实现 ActionListener 接口。

第二步：在 RandomFrame 类中定义各种需要的构件及容器，设计好布局，并实现相应功能。

第三步：在主类的 main()方法中创建 RandomFrame 类的对象，从而实现相应的功能。

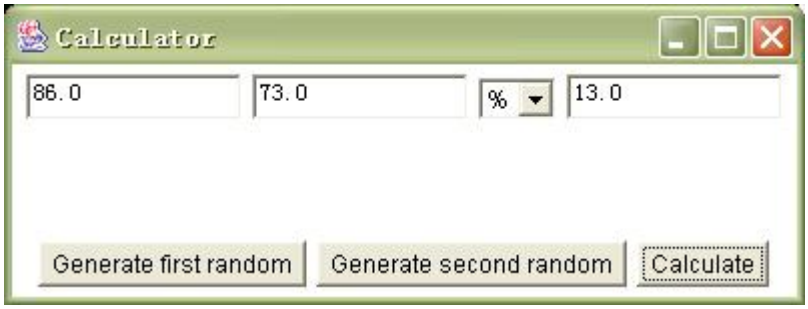
【参考程序】

```
import java.awt.*;
import java.awt.event.*;
public class X10_3_5 {
      public static void main(String[] args) {
        new RandomFrame();
      }
}
class RandomFrame extends Frame implements ActionListener{
  Button[] bt = new Button[3];
  TextField[] tf = new TextField[3];
  String[] item = {"+", "-", "*", "/", "%"};
  Choice choice;
  Panel pn1, pn2;
```

```
double d1, d2, d3;
RandomFrame(){
	int i;
	for(i=0; i<3; i++){
		bt[i] = new Button();
		bt[i].addActionListener(this);
		tf[i] = new TextField(12);
	}
	bt[0].setLabel("Generate first random");
	bt[1].setLabel("Generate second random");
	bt[2].setLabel("Calculate");
	choice = new Choice();
	for(i=0; i<item.length; i++){
		choice.addItem(item[i]);
	}
	pn1 = new Panel( );
	pn2 = new Panel( );
	pn1.add(tf[0]);
	pn1.add(tf[1]);
	pn1.add(choice);
	pn1.add(tf[2]);
	for(i=0; i<3; i++)
		pn2.add(bt[i]);
	this.add(pn1, BorderLayout.NORTH);
	this.add(pn2, BorderLayout.SOUTH);
	this.setTitle("Calculator");
	this.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e){
			System.exit(1);
		}
	} );
	this.setSize(400,150);
	this.setVisible(true);
}
public void actionPerformed(ActionEvent e){
	if(e.getSource() == bt[0]){
		d1 = (int)(90 * Math.random() )+10;
		tf[0].setText(String.valueOf(d1) );
	}
	if(e.getSource() == bt[1]){
		d2= (int)(90 * Math.random() )+10;
		tf[1].setText(String.valueOf(d2) );
	}
	if(e.getSource() == bt[2]){
```

```
                int index = choice.getSelectedIndex();
                switch (index){
                    case 0: d3 = d1 + d2; break;
                    case 1: d3 = d1 - d2; break;
                    case 2: d3 = d1 * d2; break;
                    case 3: d3 = d1 / d2; break;
                    case 4: d3 = d1 % d2; break;
                }
                tf[2].setText(String.valueOf(d3) );
            }
        }
    }
```

【运行结果】



6．编写一个菜单程序，其中包含“文本”、“格式”、“图片”和“动画”菜单，其中“文本”、“图片”和“动画”菜单中分别包含“显示文本”、“显示图片”和“播放动画”菜单项，“格式”菜单中包含“字体大小”、“字体颜色”两个菜单项，“字体大小”菜单项又包含“20”、“40”、“60”三个子菜单项，“字体颜色”菜单项又包含“红色”、“绿色”、“蓝色”三个子菜单项。当单击菜单项或子菜单项时都能实现相应功能。

【编程分析】本程序主要考察菜单的使用、文本、图片以及动画的显示等内容

第一步：建立一个类 MenuFrame，使该类继承 Frame 类，实现 ActionListener 接口。该类的主要功能是创建满足要求的 GUI（图形用户界面），并实现相应的功能。（注意：尽量用控件数组来完成，因为涉及的菜单较多，利用数组可以减少代码长度）

第二步：在主类的 main()方法中创建 MenuFrame 类的对象，从而显示窗口，完成相应的功能。

【参考程序】

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class X10_3_6 {       // 主类
    public static void main(String[] args) {
      MenuFrame mf = new MenuFrame();
    }
}
class MenuFrame extends Frame implements ActionListener{
```

```
String[] mn1_str = {"文本", "格式", "图片", "动画"}; // 以下是类的属性
String[] mn2_format_str = {"字体大小", "字体颜色"  };
String[] mi_fontSize_str = {"20", "40", "60"};
String[] mi_fontColor_str = {"红色", "绿色", "蓝色"};

MenuBar mb;
Menu[] mn1 = new Menu[4];
Menu[] mn2_format = new Menu[2] ;
MenuItem mi_0, mi_2, mi_3;
MenuItem[] mi_fontSize = new    MenuItem[3];
MenuItem[] mi_fontColor = new    MenuItem[3];

Font font;
int[] font_size = {20, 40, 60};
Color[] font_color = {Color.red, Color.green, Color.blue };
String str;
ImageIcon img;
JLabel jlb;

public MenuFrame(){    // 构造方法
      int i;
      mb = new MenuBar();
      for(i =0; i<4; i++){
            mn1[i] = new Menu(mn1_str[i]);
            mb.add(mn1[i]);
      }
      mi_0 = new MenuItem("显示文本");
      mi_0.addActionListener(this );
      mi_2 = new MenuItem("显示图片");
      mi_2.addActionListener(this );
      mi_3 = new MenuItem("播放动画");
      mi_3.addActionListener(this );
      mn1[0].add(mi_0);
      mn1[2].add(mi_2);
      mn1[3].add(mi_3);
      for(i=0; i<2; i++){
            mn2_format[i] = new Menu(mn2_format_str[i]);
            mn1[1].add(mn2_format[i]);
      }
      for(i=0; i<3; i++){
            mi_fontSize[i] = new MenuItem(mi_fontSize_str[i]);
            mi_fontSize[i].addActionListener(this );
            mn2_format[0].add(       mi_fontSize[i] );
            mi_fontColor[i] = new MenuItem(mi_fontColor_str[i]);
```

```
            mi_fontColor[i].addActionListener(this );
            mn2_format[1].add(mi_fontColor[i] );
        }

        str = "欢迎学习 Java 语言中菜单的使用！ ";
        img = new ImageIcon("img.jpg");

        jlb = new JLabel();
        jlb.setHorizontalAlignment(JLabel.CENTER);

        this.add(jlb, BorderLayout.CENTER);
        this.setTitle("Menu Frame");
        this.setMenuBar(mb);
        this.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                System.exit(0);
            }
        }    );
        this.setSize(400, 300);
        this.setVisible(true);
    }
    public void actionPerformed(ActionEvent e){     // 实现 ActionListener 接口方法
        int i;
        int currentImage = 0;
        if(e.getSource() == mi_0){
            jlb.setIcon(null);
            jlb.setText(str);
        }
        for(i=0; i<3; i++){
            if(e.getSource() == mi_fontSize[i] ){
                font = new Font("宋体",Font.PLAIN, font_size[i]);
                jlb.setFont(font);
            }
            if(e.getSource() == mi_fontColor[i]){
                jlb.setForeground(font_color[i]);
            }
        }
        if(e.getSource() == mi_2){
            jlb.setText(null);
            jlb.setIcon(img);
        }
        if(e.getSource() == mi_3){
            jlb.setIcon(null);
            jlb.setText("关于动画部分，可以利用线程去实现，读者可自行完成");
```

```
        }
    }
}
```

【运行结果】当点击不同菜单时，窗口界面将显示不同效果，以下是点击“动画->显示动画”菜单时的效果。

7．编写一个使用 JOptionPane 类对话框的程序，其中包含各种类型的 JOptionPane 对话框。

【编程分析】本程序主要考察 JOptionPane 类对话框的应用。

第一步：定义一个类 X10_3_7，该类继承 JFrame 类，实现 ActionListener 接口。在该类的窗口中添加一个按钮，当点击该按钮时，将弹出一个输入对话框，在输入对话框中输入文本，然后点击确定，此时将弹出一个确认对话框，询问输入是否正确，如果正确，点击“是”按钮，此时弹出消息对话框，说明你输入的内容正确，否则点击“否”按钮，此时也弹出一个消息对话框，说明你输入的内容错误。

第二步：在 main()方法中创建本类对象，设置窗口大小并显示窗口。

【参考程序】

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class X10_3_7 extends JFrame implements ActionListener{
    JLabel jlb;
    JButton jbt;
    public static void main(String args[]){
        X10_3_7 f = new X10_3_7 ();
        f.setTitle("JOptionPane 对象的应用");
        f.setSize(250,135);
        f.setVisible(true);
    }
   public X10_3_7 ()     {
```

```
            this.setFont(new Font("",Font.BOLD,40));
            Container ct = getContentPane();
            jlb = new JLabel("JOptionPane 应用举例",JLabel.CENTER);
            jbt = new JButton("输入姓名");
            jbt.addActionListener(this);
            ct.add(jlb,BorderLayout.CENTER);
            ct.add(jbt,BorderLayout.SOUTH);
            addWindowListener(new WindowAdapter() {
                    public void windowClosing(WindowEvent e) { System.exit(0); }
                } );
        }
        public void actionPerformed(ActionEvent e){
            X10_3_7 obj = new X10_3_7 ();
            String result = JOptionPane.showInputDialog("请输入姓名");
            if(result != null){
                int x = JOptionPane.showConfirmDialog(obj, "你的名字是"+result+"吗?");
                if (x == 0)
                    JOptionPane.showMessageDialog(obj, "你的名字是"+result,
                                        "MessageDialog", JOptionPane.OK_OPTION);
                if(x == 1)
                    JOptionPane.showMessageDialog(obj, "输入错误！","MessageDialog",
                                                JOptionPane.ERROR_MESSAGE);
            }
        }
    }
```

【运行结果】

第 11 章

一、填空题

1．URL 是 统一资源定位器 的简称，它表示 Internet/Intranet 上的资源位置。这些资源可以是一个文件、一个 目录 或一个 对象 。

2．每个完整的 URL 由四部分组成： 传输协议 、 主机名 、 端口号 以及 文件名称 。

3．两个程序之间只有在 地址 和 端口号 方面都达成一致时才能建立连接。

4．使用 URL 类可以简单方便地获取信息，但是如果希望在获取信息的同时，还能够向远方的计算机节点传送信息，就需要使用另一个系统类库中的类 URLConnection 。

5．Socket 称为 套接字 ，也有人称为“插座”。在两台计算机上运行的两个程序之间有一个双向通信的链接点，而这个双向链路的每一端就称为一个 Socket 。

6．Java.net 中提供了两个类： ServerSocket 和 Socket ，它们分别用于服务器端和客户端的 socket 通信。

7. URL 和 Socket 通信是一种面向 连接 的流式套接字通信，采用的协议是 TCP 协议。UDP 通信是一种 无连接 的数据报通信，采用的协议是数据报通信协议 UDP 。

8．Java.net 软件包中的类 DatagramSocket 和类 DatagramPacket 为实现 UDP 通信提供了支持。

9． receive() 和 send() 是 DatagramSocket 类中用来实现数据报传送和接收的两个重要方法。

10．JDBC API 提供的类和接口在 java.sql 包中定义。

二、编程题

1．利用 URL 读取网络上的资源（假定网页或文本文件）并将该资源保存到指定的文件中。
【提示】参照 EX11_2 程序来完成。

2．利用 URLConnection 类对象与 URL 地址指定的远程节点建立一条 HTTP 协议的连接通路，下载 URL 定位资源中的网页并保存到当前路径中。
【提示】参照课本“例 11.3”程序来完成。

3．编写一个简单的客户机/服务器程序，通过客户机向服务器指定端口发送一段信息。服务器一直处于等待状态直到接收到客户端发送的信息后输出该信息并关闭端口和连接，结束程序的运行，而客户端也结束运行。
【提示】参照课本“例 11.6”和“例 11.7”程序来完成。

4．编写一个简单的客户机/服务器程序，通过客户机向服务器发送信息，而服务器等待客户机发送信息。当客户端向服务器发送信息后服务器接收信息并输出该信息，直到接收到“END”字符后，服务器端向客户端发送“Server received data end”信息并关闭端口和连接，结束服务器端程序的运行，同时客户端也结束程序运行。
【提示】参照课本“例 11.6”和“例 11.7”程序来完成。