**一、简答题**

1.什么是字节码，采用字节码的好处是什么

字节码：Java源代码经过虚拟机编译器编译后产生的文件（即扩展为.class的文件），它不面向任何特定的处理器，只面向虚拟机。

采用字节码的好处：Java语言通过字节码的方式，在一定程度上解决了传统解释型语言执行效率低的问题，同时又保留了解释型语言可移植的特点。所以Java程序运行时比较高效，而且，由于字节码并不专对一种特定的机器，因此，Java程序无须重新编译便可在多种不同的计算机上运行。

2.什么是平台无关性，Java语言是如何实现平台无关性的

平台无关性就是一种语言在计算机上的运行不受平台的约束，一次编译，到处执行。用Java创建的可执行二进制程序，能够不加改变的运行于多个平台。

3. Java语言程序有几种？他们包含哪几个方面？

应用程序和小程序。一个是完整的程序，主类包含了主方法，另一个是嵌入在HTML文档中的JAVA程序，需要搭配浏览器  
4.什么是Java程序的主类？应用程序与小程序的主类有何不同？  
 包含了main()方法的是主类。应用程序的主类不一定是public类，但小程序主类一定是Public类。

5. 什么是JDK？什么是JRE？JDK与JRE的关系是什么？

1.由JAVA API,JAVA运行环境和一组建立，测试工具的JAVA实用程序等组成。同时并提供JRE

2.有JAVA运行环境，但无项目开发功能

3.JDK包含 JRE

6. 环境变量Path和ClassPath的作用是什么？如何设置Path环境变量？

作用:通过配置path环境变量，将javac指令所在目录也就是JDK安装目录下的bin目录配置到path变量下，即可使javac指令在任意目录下运行。使classpath目录中的.class文件可以在任意目录运行。

7.Java应用程序与小程序之间有哪些差别

运行方式，函数不同，入口函数不同

8.类与对象的区别

对象是具有类类型的变量。类是对象的抽象，而对象是类的具体实例。类是抽象的，不占用内存，而对象是具体的，占用存储空间。 类是用于创建对象的蓝图，它是一个定义包括在特定类型的对象中的方法和变量的软件模板

9.如何定义一个类，类的结构是怎样的

修饰词+class+类名+{ ，一个类中通常都包含数据与函数两种类型的元素，我们一般把它叫作属性和成员函数。

10.定义一个类时所使用的修饰符有哪几个，作用分别是，是否可以混用

定义一个类时所使用的修饰符有 4 个，一个类可以有多个修饰符，且无先后顺序之分，但 abstract 和 final 相互独立，所以不能同时应用在一个类的定义中，每个修饰符的作用如下：

Public：将一个类声明为公共类，它可以被任何对象访问

Abstract：将一个类声明为抽象类，没有实现方法，需要子类提供方法的实现，所以不能创建该类的实例

Final：将一个类声明为最终类即非继承类，表示它不能被其他类所继承

缺省：缺省修饰符时，则表示只有在相同包中的对象才能使用这样的类

可以混用，但是abstract和final不能混用

11.成员变量与局部变量的区别有哪些

1、在类中的位置不同

成员变量：在类中方法外面

局部变量：在方法或者代码块中，或者方法的声明上（即在参数列表中）

2、在内存中的位置不同，可以看看Java程序内存的简单分析

成员变量：在堆中（方法区中的静态区）

局部变量：在栈中

3、生命周期不同

成员变量：随着对象的创建而存在，随着对象的消失而消失

局部变量：随着方法的调用或者代码块的执行而存在，随着方法的调用完毕或者代码块的执行完毕而消失

4、初始值

成员变量：有默认初始值

局部变量：没有默认初始值，使用之前需要赋值，否则编译器会报错

12.什么是方法的重载

方法重载是指在一个 类 中定义多个同名的方法，但要求每个方法具有不同的 参数 的类型或参数的个数。

13.构造方法有那些特性

1.当没有指定构造方法时，系统会自动添加无参的构造方法。 2.构造方法可以重载：方法名相同，但参数不同的多个方法，调用时会自动根据不同的参数选择相应的方法。 3.构造方法是不被继承的

14.对象的相等与指向它们的引用相等，两者有何不同

对象的相等指的是对象的内容相等；

指向它们的引用相等指的是引用变量指向的地址相同；

15. 什么是静态初始化器，其作用是什么？静态初始化器由谁在何时执行？它与构造方法有何不同？

静态初始化器是由关键字 static 修饰的一对大括号“{}”括起来的语句组。它是用来完成初始化工作的；静态初始化器有 Java 虚拟机在类初始化的时候一次执行；

静态初始化器与构造方法有以下几点不同：

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

2. 构造方法是在用 new 运算符创建新对象时由系统自动执行，而静态初始化器一般不能由程序来调用，它是在所属的类被加载入内存时由系统调用执行的。

3. 用 new 运算符创建多少个新对象，构造方法就被调用多少次，但静态初始化器则在类被加载入内存时只执行一次，与创建多少个对象无关。

4. 不同于构造方法，静态初始化器不是方法，因而没有方法名，返回值和参数。

16. 什么是“多态”机制？Java 语言中是如何实现多态的？

多态是指“一种定义，多种实现”，多态有两种表现形式：重载和覆盖；多态是由方法重载，方法重写，继承，自动转型等引起的一系列特性。

17. 方法的“覆盖”与方法的“重载”有何不同？

重载是指在同一个类，内定义多个名称相同，但参数个数或类型不同的方法，从而，Java 系统便可根据参数的个数或类型，调用相对应的方法；

覆盖是指在子类中，定义名称、参数个数与类型均与父类完全相同的方法，用来重写父类中同名方法的

18. this 和 super 分别有什么特殊含义？

super 是从子类调用调用父类的成员，包括构造方法、成员变量和成员方法。

This 是用来调用同一类内的成员，包括构造方法、成员变量和成员方法。

19. 什么是抽象类与抽象方法？使用时应注意哪些问题？

抽象类是以修饰符 abstract 修饰的类，抽象方法是以 abstract 关键字开头的方法，此方法只声明返回值的数据类型、方法名称与所需要的参数，但没有方法体。

需要注意的是：

1. 由于抽象类是需要被继承的，所以 abstract 类不能用 final 来修饰。也就是说，一个类不能既是最终类，又是抽象类，即关键字 abstract 与 final 不能合用。

2. abstract 不能与 private、static、final 或 native 并列修饰同一方法。

20. 什么是接口？为什么要定义接口？

Java接口是一系列方法的声明，是一些方法特征的集合，一个接口只有方法的特征没有方法的实现，因此这些方法可以在不同的地方被不同的类实现，而这些实现可以具有不同的行功能;

21. 如何定义接口？接口与抽象类有哪些异同？

接口与抽象类有以下不同：

1. 接口的数据成员都是静态的，并且必须初始化。

2. 接口中的方法必须全部都声明为 abstract 的，也就是说，接口不能像抽象类一样拥有一般的方法，而必须全部是抽象方法。

接口定义的语法格式如下：

[public] interface 接口名称 [extends 父接口列表]

{

[public] [static] [final] 数据类型 成员变量名 = 常量；

[public] [abstract] 返回值的数据类型 方法名(参数表)；

}

22. 内部类的类型有几种？分别在什么情况下使用？它们所起的作用有哪些？

内部类分为： 成员内部类、局部内部类、静态嵌套类、匿名内部类 ；内部类的主要作用是将逻辑上相关的类放到一起；而匿名类是一种特殊的内部类，它没有类名，在定义类的同时，就生成该类的一个实例，由于不会在其他地方用到该类，所以不用取名字，因而又被称为匿名内部类。

23. 什么是异常？简述 Java 语言的异常处理机制。

异常是指在程序运行中由代码产生的一种错误；

Java 语言的异常处理机制：

1. 发现异常2. 抛出异常3. 捕获异常

24. Throwable 类的两个直接子类 Error 和 Exception 的功能各是什么？用户可以捕获到的异常是哪个类的异常？

Error 类及其子类的对象，代表了程序运行时 Java 系统内部的错误。

Exception 子类则是供应用程序使用的，它是用户程序能够捕捉到的异常情况；用户可以捕捉的异常是 Exception 异常。

25. Exception 类有何作用？每个 Exception 类的对象代表什么？

Exception 类对象是 Java 程序抛出和处理的对象，它有各种不同的子类分别对应各种不同类型的异常。

26. 什么是运行时异常？什么是非运行时异常？

Exception 类中的一个子类 RuntimeException 代表运行时异常，它是程序运行时自动地对某些错误做出反应而产生的；

除 RuntimeException 之外，其它则是非运行时异常，这种异常经常是在程序运行过程中由环境原因造成的异常。

27. 抛出异常有哪两种方式？

1. 系统自动抛出的异常。

2. 指定方法抛出异常

28. 若 try 语句结构中有多个 catch() 子句，这些子句的排列顺序与程序执行效果是否有关？为什么？

有关；由于异常对象与 catch 块的匹配是按照 catch 块的先后排列顺序进行的。若将子类异常的 c atch 语句块放在父类异常 catch 语句块的后面，则编译不能通过。

29. 什么是文件的输入与输出？

向文件写入数据；从文件读取数据

30.什么是流？Java语言中分为哪两种流？这两种流有何差异？

流是指计算机各部件之间的数据流动。

按照数据的传输方向，流可分为输入流与输出流。

输入流只能从其读取数据而不能向其写入数据；

输出流只能向其写入数据而不能从其读取数据。

31.InputStream、OutputStream、Reader和Writer四个类在功能上有何异同？

InputStream和OutputStream类通常是用来处理位流，也就是二进制文件，但也可以处理纯文本文件；

Reader和Writer类是用来处理字符流，也就是纯文本文件。

32. 在泛型中，类型通配符的主要作用是什么？

无边界的通配符的主要作用就是让泛型能够接受未知类型的数据；使用固定上边界的通配符的泛型, 就能够接受指定类及其子类类型的数据. 使用固定下边界的通配符的泛型, 就能够接受指定类及其父类类型的数据

33.简述线程的基本概念，程序、进程、线程的关系是什么？

运⾏⼀个进程时，将程序中⼀些彼此相对独⽴的代码段可以同时运⾏，从⽽获得更⾼的执⾏ 效率的运⾏单位，称为线程。 程序是含有指令和数据的⽂件，被存储在磁盘或其他的数据存储设备中，是静态的代码； 进程是程序的⼀次执⾏过程，是系统运⾏程序的基本单位，进程是动态的。 系统运⾏⼀个程序即是⼀个进程从创建、运⾏到消亡的过程。 线程与进程相似，也是⼀个执⾏中的程序，但线程是⼀个⽐进程更⼩的执⾏单位。⼀个进程 在其执⾏过程中可以产⽣多个线程，形成多条执⾏线路。

34.什么是多线程？为什么程序的多线程功能是必要的？

多线程就是同时执⾏⼀个以上的线程，⼀个线程的执⾏不必等待另⼀个线程执⾏完才执⾏，所有的 线程都可能发⽣在同⼀时刻。但操作系统并没有将多个线程看做多个独⽴的应⽤，来实现线程的调 度和管理以及资源分配。 由于可以共享内存，所以各线程之间的通信速度很快，线程之间进⾏切换所占⽤的系统资源也较少

35.多线程与多任务的差异是什么？

多任务是针对操作系统⽽⾔，表示操作系统可以同时运⾏多个应⽤程序； 多线程是指⼀个程序⽽⾔的，表示在⼀个程序内部可以同时执⾏多个线程。

36. Java程序实现多线程有哪两个途径？ Thread类和Runnable接⼝。

1. 什么是平台无关性？JAVA语言是如何实现平台无关性的？
2. 自动类型转换的前提是什么？简述转换时的优先级顺序。
3. 简述JAVA程序中的成员变量和局部变量的区别。
4. 简述JAVA程序中对象的相等和指向它们的引用相等的区别。
5. 什么是接口？简述接口与抽象类的区别。
6. 什么是流？JAVA语言中分为哪两种流？简述这两种流的区别。
7. Java实现多线程有哪两种途径？有什么区别？
8. 简述 Java应用程序与Java小程序的区别？
9. 简述基本类型变量和引用类型变量的区别（举例说明）。
10. 简述JDK与JRE的关系。
11. 简述方法的重载与覆盖的区别。
12. 什么是接口？为什么要定义接口？
13. InputStream、OutputStream、Reader和Writer四个类的区别？
14. 简述Java的运行机制。
15. 什么是构造方法?Java 中构造方法须同时满足的条件。
16. 什么是方法的覆盖？什么是方法的重载？两者有什么区别？
17. 简述什么是集合？并列举集合中常用的类和接口
18. 什么是流？Java语言中分为哪两种流？这两种流有何区别？
19. 简述JDBC中Driver、Connection、Statement和ResultSet的功能。
20. 用Java程序实现流式Socket通信，需要使用哪两个类？它们是如何定义的？
21. 简述JRE和JDK的区别。
22. 简述Java语言的八种基本数据类型，并说明每种数据类型所占用的空间大小？
23. 什么是多态？
24. 简述抽象类和接口的区别？
25. 什么是自动装箱和自动拆箱？
26. 简述字节流和字符流的区别？
27. 简述JDBC编程步骤？
28. 简述TCP协议和UDP协议？

**二、阅读分析题**

**1. 指出下面程序的错误语句及原因**

public class TestDataConvert {

public static void main(String[] args) {

int i=1, j;

float f1=0.1;

i = i \* 0.1;

float f2=1234;

long s2=88888888888;

double d1 = 1e20, d2=125;

byte b1 = 10, b2 = 12, b3 = 131;

j = j+10;

byte b = b1+b2;

}

}

1） float f1=0.1;；原因浮点默认double，无法直接赋值，需要后面加上f；

2） i=i\*0.1 ；原因 0.1为浮点型，会损失精度，应该转为（int） ；

3） float f2=1234; ；原因 1234为int，无法直接赋值给float，应写为f2=1234f；

4）long s2=88888888888;；原因这么多8超出了int型，应该写为s2=888888888L；

5）byte b=b1+b2；原因两个byte相加结果为int，无法给byte赋值，需要进行类型转换，byte b=(byte)(b1+b2)；

**2. 请给出下列程序的运行结果并给出原因**

public class TestStrEqual {

public static void main(String[] args) {

String str1 = new String("Chinese");

String str2 = new String("Chinese ");

System.out.println(str1 == str2); 输出结果： false ；

原 因：new了两个不同地址

String str3 = "HelloWorld";

String str4 = "HelloWorld";

System.out.println(str3 == str4); 输出结果： ture ；

原 因：str3和str4都是字符串常量，编译器在编译时会将相同的字符串常量引用同一个对象，所以它们引用的是同一个对象，使用==比较的是引用地址，所以结果为true。

System.out.println(str3.equals(str4)); }} 输出结果： ture ；

原 因：equals()方法比较的是字符串的内容，而不是引用地址。str3和str4的内容相同，所以结果为true。

**3. 请给出下列程序的运行结果并给出原因**

public class TestThis {

int i = 0;

TestThis increment() {

i++;

return this;

}

void print() {

System.out.println("i = " + i);

}

public static void main(String[] args) {

TestThis testThis = new TestThis(100);

testThis.increment().increment().increment().print();

}

}

输出结果：无法运行；

原 因：没有参数为int的构造类型

**4.分析语句1）与语句2)-7)是否构成重载并说明原因。**

1) int add(int x, int y);

2) int add(int x, int y, int z);

3) float add(float f1, float f2);

4) float add(float f1, int y);

5) float add(int y, float f1);

6) float add(int x, int y);

7) int add(int u, int v);

结果：构成重载；

原 因：方法签名不同

**5. 请给出下列程序的运行结果并给出原因**

class a{

public a(){

System.out.println("aaa");

}

}

public class Extends extends a {

Extends(){

this(12);

System.out.println("bbb");

}

Extends(int x){

System.out.println("ccc");

}

public static void main(String[] args)

{

new Extends();

}

}

输出结果：aaa

ccc

bbb；

原 因：在main方法中创建了一个Extends对象。

由于Extends类继承了a类，因此首先会调用a类的无参构造方法，输出aaa。

然后，在Extends类的无参构造方法中，使用this(12);调用了带有一个整数参数的构造方法。

带有一个整数参数的构造方法输出ccc。

最后，在Extends类的无参构造方法中，输出bbb。

因此，输出结果为aaa，ccc，bbb。

**6.下面是一个类的定义，请将其补充完整**。

public class \_\_\_\_①\_\_\_ {

String name;

int age;

char gender;

String address;

}

public class MainClass{

public static void main(String[] args) {

Person person=\_\_\_\_\_\_②\_\_\_\_\_;

person.name=＂张三＂;

person.age=22;

person.gender='男';

person.address=" 36 Pengjiaping Road, Qilihe District, Lanzhou, Gansu, China ";

}}

① Person；

②new Person；

**7. 设下列try catch 语句块中的第二个语句st2将引起一个异常，试回答下列问题**

try{

st1; st2; st3;

}catch(ExceptionType e1){}

catch(ExceptionType e2){}

finally{ st4; }

st5;

（1）st3会执行吗？ 如果st2引发了一个异常，并且没有在catch子句中被捕获，那么st3将不会执行。程序将跳过st3并继续执行catch子句或finally块（如果存在的话）。；

(2) 如果catch子句捕获了异常，st4会执行吗？无论异常是否被捕获，finally块中的语句（包括st4）都会执行。即使异常被捕获并且相应的catch子句被执行，finally块仍然会在catch子句执行完毕后执行。；

（3）如果异常被捕获，st5会被执行吗？如果异常在try块内被捕获并由相应的catch子句处理，那么程序将继续执行try-catch语句块之后的语句。因此，st5会被执行。如果异常没有被捕获，程序将在抛出异常后立即终止，st5将不会被执行。

**8.****以下程序有三处错误，请仔细阅读，找出错误并说明原因**。

abstract class AbstractA{

abstract void f(){};

public abstract void k();

}

class AbstractB extends AbstractA {

protected void f(){ }

void k() { System.out.print("I am subclass"); }

public static void main(String[] args) {

AbstractA a=new AbstractA ();

a.f();

a.k();

}

}

① ；

② ；

③

根据提供的代码，以下是对每个错误的分析和解释：

① 错误：抽象类`AbstractA`不能被实例化。

解释：抽象类是不能直接实例化的，只能被子类继承并实现其抽象方法。在`main`方法中，尝试实例化`AbstractA`是不允许的。

修正：将 `AbstractA a = new AbstractA();` 修改为 `AbstractA a = new AbstractB();`，使用`AbstractB`的实例来引用抽象类。

② 错误：抽象方法`k()`在`AbstractA`中没有实现。

解释：在抽象类中声明的抽象方法必须在具体的子类中进行实现。在`AbstractA`中声明了抽象方法`k()`，但在`AbstractB`中没有提供实现。

修正：在`AbstractB`类中添加实现抽象方法`k()`，如下所示：

```java

void k() {

System.out.print("I am subclass");

}

```

③ 错误：抽象方法`f()`的访问修饰符不允许为默认（package-private）。

解释：抽象方法在抽象类中必须有访问修饰符，不能使用默认（package-private）访问修饰符。在`AbstractA`中，抽象方法`f()`的访问修饰符是默认的，这是不正确的。

修正：将 `abstract void f(){};` 修改为 `public abstract void f();`，或者根据需求选择合适的访问修饰符（public、protected等）。

修正后的代码如下所示：

```java

abstract class AbstractA {

public abstract void f();

public abstract void k();

}

class AbstractB extends AbstractA {

protected void f() {

}

public void k() {

System.out.print("I am subclass");

}

public static void main(String[] args) {

AbstractA a = new AbstractB();

a.f();

a.k();

}

}

```

修正后的代码已经解决了原先存在的错误。

**9. 设下列try catch 语句块中的第一个语句s1引起一个异常，并将被第一个catch捕获到，试回答下列问题。**

try{

s1;

s2;

s3;

}catch(ExceptionType e1){

}

catch(ExceptionType e2){

}

s4;

1. s2会执行吗？

（2）如果异常未被捕获，s4会被执行吗？

（3）如果catch子句捕获了异常，s4会执行吗？

**10. 请给出下列程序的运行结果并给出原因**

public class stringT {

public static void main(String[] args) {

String s1 = "Hello"; String s2 = "World";

String s3 = "Hello";

System.out.println(s1 == s3); 运行结果： ；

原 因：

s1 = new String("Hello");

s2 = new String("Hello");

System.out.println(s1 == s2); 运行结果： ；

原 因：

System.out.println(s1.equals(s2)); 运行结果： ；

原 因：

}

}

以下是对每个输出结果的分析和解释：

1) `System.out.println(s1 == s3);`

输出结果：`true`

原因：在Java中，字符串常量池会对相同的字符串进行重用，因此`s1`和`s3`指向的是同一个字符串对象。

2) `System.out.println(s1 == s2);`

输出结果：`false`

原因：使用`new`关键字创建字符串对象时，会在堆内存中分配新的内存空间，因此`s1`和`s2`指向的是不同的字符串对象。

3) `System.out.println(s1.equals(s2));`

输出结果：`true`

原因：`equals()`方法比较的是字符串对象的内容，而不是引用地址。在这里，虽然`s1`和`s2`引用的是不同的字符串对象，但它们的内容是相同的，因此返回`true`。

因此，程序的运行结果是：

```

true

false

true

```

**11. 请给出下列程序的运行结果并给出原因**。（不确定）

public class Test {

public static void main(String[] args){

int j=12345678;

String s = String.valueOf(j);

运行结果 可能是8 ；

System.out.println(s.length());

String s = "Jerry,F,2005";

String[] sPlit=s.split(",");

for(int i=0;i<sPlit.length;i++){

System.out.println(sPlit[i]);

运行结果 Jerry ；

F ；

2005 ；

}

}

}

**12. 请给出下列程序的运行结果并给出原因**。

public class Leaf {

int i = 0;

Leaf increment() {

i++;

return this;

}

void print() {

System.out.println("i = " + i);

}

public static void main(String[] args) {

Leaf x = new Leaf();

x.increment().increment().increment().print();

}

}

运行结果：i=3；

原 因：在该程序中，Leaf类定义了一个实例变量 i 和两个方法 increment() 和 print()。

在 main() 方法中，首先创建了一个 Leaf 类的对象 x，然后通过 x.increment().increment().increment().print() 这一连续的调用链对 i 进行了三次递增操

1. 阅读以下程序段，找出错误并改正

public class Test1{

public static void main(String args[]){

char c=65535;

byte b=127;

int height=100;

float f=3.14

}

}

修改：float f=3.14f;

13阅读以下程序段，找出错误并改正

public interface Com{

abstract void cry();

pubic int x=0;

static int y=0;

}

abstract class Cat implement Com{

abstract void cry();

}

在给定的程序段中，存在以下问题：

1. 拼写错误：在接口 `Com` 的方法声明中，`public` 关键字拼写错误为 `pubic`。

2. 接口字段修饰符错误：接口 `Com` 中的字段 `x` 的修饰符应为 `public static`，而不是 `abstract`。

3. 类名拼写错误：抽象类 `Cat` 的关键字拼写错误为 `implement`，应改为 `implements`。

4. 方法声明错误：抽象类 `Cat` 实现接口 `Com` 后，应实现接口中的抽象方法 `cry()`。

以下是修正后的代码：

```java

public interface Com {

void cry();

public int x = 0;

static int y = 0;

}

abstract class Cat implements Com {

public abstract void cry();

}

```

修正后的代码已经改正了拼写错误，并修正了接口字段的修饰符和抽象类中方法的声明。

14阅读以下程序，给出程序运行结果

public class E{

public static void main(String args[]){

int[] a={10,20,30,40},b[]={{1,2},{4,5,6,7}};

b[0]=a;

b[0][1]=b[1][3];

System.out.println(b[0][1]);

System.out.println(a[1]);

}

}

给定的程序运行结果如下：

```

7

20

```

解释：

你所给出的代码没有错误，也没有需要修改的地方。

这段程序的输出结果为：7 7。

在这段程序中，变量 `a` 是一个长度为4的一维数组，`b` 是一个二维数组，其第一行包含两个元素1和2，第二行包含四个元素4、5、6和7。执行语句 `b[0]=a;` 之后，将 `a` 的引用赋值给了 `b[0]`，因此 `b[0]` 指向了 `a` 所表示的数组对象。

接着执行语句 `b[0][1]=b[1][3];`，将 `b` 数组的第一行第二列元素的值改为了 `b` 数组的第二行第四列元素的值，即 b[0][1] 的值变成了 7。由于 `b[0]` 和 `a` 引用同一个数组对象，所以 `a[1]` 的值也被改为了 7。

因此程序输出的结果分别是 `7` 和 `7`。

15阅读以下程序，给出程序运行结果

public class Test {

int n;

static int sum=0;

void setN(int n){this.n=n;}

int getSum(){

for(int i=1;i<=n;i++){

sum+=i;

}

return sum;

}

public static void main(String[] args) {

Test test1=new Test();

Test test2=new Test();

test1.setN(3);

test2.setN(5);

int s1=test1.getSum();

int s2=test2.getSum();

System.out.println("s1+s2="+(s1+s2));

}

}

给定的程序运行结果如下：

```

s1+s2=27

```

解释：

static int *sum*=0;

调用两次getSum方法，得到27。

16阅读以下程序，给出程序运行结果

import java.io.IOException;

public class E{

public static void main(String args[]){

try{ method();

}catch(IOException e){

System.out.printf("你好");}

finally{

System.out.println("fine thanks"); }

}

public static void method() throws IOException{

throw new IOException();

}

}

给定的程序运行结果如下：

```

你好

fine thanks

```

解释：

1. 定义了一个名为 `E` 的类。

2. 在 `main` 方法中，调用了 `method` 方法。

3. `method` 方法声明了可能抛出 `IOException` 异常。

4. 在 `main` 方法中使用 `try-catch` 块捕获 `IOException` 异常，并在 `catch` 块中输出字符串 "你好"。

5. 无论是否发生异常，`finally` 块中的代码都会被执行，输出字符串 "fine thanks"。

6. 由于 `method` 方法抛出了 `IOException` 异常，所以程序会执行 `catch` 块中的代码，输出 "你好"。

7. 最后执行 `finally` 块中的代码，输出 "fine thanks"。

17 阅读以下程序，给出程序运行结果

import java.util.\*;

public class Test {

public static void main(String[] args) {

LinkedList<Integer> list=new LinkedList<Integer>();

for(int k=1;k<=10;k++)

list.add(new Integer(k));

list.remove(5);

list.remove(5);

Integer m=list.get(5);

System.out.println(m.intValue());

}

}

给定的程序运行结果如下：

```

8

```

解释：

1. 导入了 `java.util` 包。

2. 定义了一个名为 `Test` 的类。

3. 在 `main` 方法中创建了一个 `LinkedList<Integer>` 类型的对象 `list`。

4. 使用 `for` 循环将整数从 1 到 10 添加到 `list` 中。

5. 通过调用 `list.remove(5)` 两次，从 `list` 中移除索引为 5 的元素。

6. 使用 `list.get(5)` 获取索引为 5 的元素，即第六个元素。

7. 使用 `m.intValue()` 将 `Integer` 对象 `m` 转换为基本类型 `int`。

8. 最后输出结果为 8，表示第六个元素的值。

1. 以下为求解1到100的倒数和的程序，找出程序中的错误并改正

public class Test{

public static void main(String args[]){

double sum=0.0;

for(int i=1;i<=100;i++)

sum+=1/i;

System.out.println("sum="+sum);

}

}

在给定的程序中，有一个逻辑错误。在循环中，将整数 `i` 作为分母进行除法运算，但是整数除法的结果会被截断为整数，导致结果不准确。为了修正这个错误，可以将分母 `i` 转换为浮点数类型，使得除法运算得到准确的结果。

以下是修正后的程序：

```java

public class Test {

public static void main(String[] args) {

double sum = 0.0;

for (int i = 1; i <= 100; i++) {

sum += 1.0 / i;

}

System.out.println("sum=" + sum);

}

}

```

修正后的程序使用 `1.0 / i` 将整数 `i` 转换为浮点数，确保了除法运算的准确性。

19 阅读以下程序段，找出错误并改正

public class Test implements Thread{

public static void main(String args[]){

Test t = new Test();

Thread tt = new Thread(t);

tt.start();

}

public void run(){

for(int i=0;i<5;i++){

System.out.println("i="+i);

}

}

}

在给定的程序段中，存在两个错误：

1. `Test` 类不能直接实现 `Thread` 接口，因为 `Thread` 是一个具体类而不是接口。正确的做法是让 `Test` 类继承 `Thread` 类。

2. 在创建 `Thread` 对象时，应该传递 `Runnable` 对象作为参数，而不是直接传递 `Test` 对象。因此，需要将 `Test` 类实现 `Runnable` 接口，并在创建 `Thread` 对象时将其作为参数传递。

以下是修改后的程序：

```java

public class Test extends Thread {

public static void main(String args[]) {

Test t = new Test();

t.start();

}

public void run() {

for (int i = 0; i < 5; i++) {

System.out.println("i=" + i);

}

}

}

```

修正后的程序将 `Test` 类继承自 `Thread` 类，并在 `main` 方法中创建 `Test` 对象并调用 `start` 方法，启动线程执行 `run` 方法中的代码。

20 阅读以下程序，给出程序运行结果

public class Test {

public static void main(String args[]) {

for(int i=1;i<=4;i++) {

switch(i) {

case 2:

System.out.print("B");

case 3:

System.out.print("C");

case 1:

System.out.print("A");

case 4:

System.out.print("D");

break;

}

}

}

}

程序的运行结果是：ABCD

原因是在 `switch` 语句中，每个 `case` 后面没有显式地使用 `break` 关键字来终止 `switch` 语句的执行。在 `switch` 中，如果没有 `break` 语句，程序将会继续执行后续的 `case` 语句，直到遇到 `break` 或者 `switch` 语句结束。

在给定的程序中，当 `i` 的值为 2 时，首先输出 "B"，然后由于没有 `break` 语句，程序继续执行下一个 `case` 语句，输出 "C"。同样地，当 `i` 的值为 1 时，输出 "A"，当 `i` 的值为 4 时，输出 "D"。

因此，程序会按照顺序输出 "ABCD"。

21阅读以下程序，给出程序运行结果

class Father{

void speak(){

System.out.println("I am father!");

}

}

public class Son extends Father{

void speak(){

super.speak();

System.out.println("I am son!");

}

public static void main(String args[]){

Son cxz=new Son();

cxz.speak();

}

}

程序的运行结果是：

```

I am father!

I am son!

```

在给定的程序中，定义了一个父类 `Father` 和一个子类 `Son`。子类 `Son` 继承了父类 `Father`。

在 `Son` 类中，重写了父类的 `speak()` 方法。在子类的 `speak()` 方法中，首先调用了父类的 `speak()` 方法，使用 `super.speak()` 实现。然后输出 "I am son!"。

在 `main()` 方法中，创建了一个 `Son` 类的对象 `cxz`，并调用其 `speak()` 方法。

因此，程序的输出结果为：

```

I am father!

I am son!

```

22 阅读以下程序，给出程序运行结果

public class Test{

public static void main(String args[]){

try{

method();

}catch(IOException e){

System.out.printf("程序运行出错，");

}finally{

System.out.println("谢谢使用。");

}

}

public static void method() throws IOException{

throw new IOException();

}

}

程序的运行结果是：

```

程序运行出错，谢谢使用。

```

在给定的程序中，定义了一个 `Test` 类，并在 `main` 方法中调用了 `method` 方法。

`method` 方法声明了 `throws IOException`，表示该方法可能会抛出 `IOException` 异常。

在 `main` 方法中，调用了 `method` 方法，并使用 `try-catch` 块捕获了 `IOException` 异常。在 `catch` 块中输出了 "程序运行出错，"。

无论是否发生异常，`finally` 块中的代码都会被执行。在 `finally` 块中输出了 "谢谢使用。"。

由于 `method` 方法中抛出了 `IOException` 异常，并且在 `main` 方法中捕获了该异常，所以程序会执行 `catch` 块中的代码。

因此，程序的输出结果为：

```

程序运行出错，谢谢使用。

```

23 阅读以下程序，给出程序完成的功能描述

public class Test {

public static void main(String args[]) {

String[] labels = {"1","2","3","4","5","6"};

JFrame fr = new JFrame();

fr.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

fr.setLayout(new GridLayout(1,6));

for(int i=0;i < labels.length;i++)

fr.add(new Button(labels[i]));

fr.pack();

fr.setVisible(true);

}

}

给定的程序创建了一个包含6个按钮的窗口界面。

程序首先创建了一个字符串数组 `labels`，其中包含了6个标签。

接下来，创建了一个 `JFrame` 对象 `fr`，并设置其默认的关闭操作为 `JFrame.EXIT\_ON\_CLOSE`，即关闭窗口时退出程序。

然后，设置 `fr` 的布局管理器为 `GridLayout`，其中指定了1行6列的网格布局。

接下来，使用循环将6个按钮添加到 `fr` 中，每个按钮使用数组 `labels` 中的一个标签。

调用 `fr.pack()` 方法调整 `fr` 的大小以适应所有组件。

最后，设置 `fr` 可见，即显示窗口。

因此，该程序的功能是创建一个窗口界面，其中包含了6个按钮，并且按照网格布局进行排列。

**三、程序设计**

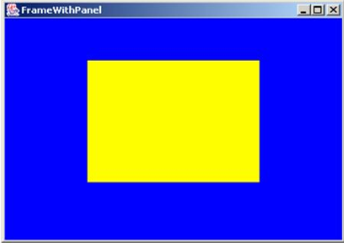
1. 编写实现将命令行中输入的数据存入数组中，并按从小到大的顺序进行排序。
2. 以m行n列的二维数组为参数进行方法调用，分别计算二维数组各列元素之和，返回并输出所计算的结果。
3. 已有若干学生的数据，包括学号、姓名、成绩，要求输出这些学生的数据并计算出学生人数和平均成绩（学生人数和总成绩用静态数据成员表示）

public class StudentData {  
 private String studentId;  
 private String studentName;  
 private double score;  
  
 private static int studentCount ;  
 private static double totalScore ;  
  
 public StudentData(String studentId, String studentName, double score) {  
 this.studentId = studentId;  
 this.studentName = studentName;  
 this.score = score;  
  
 studentCount++;  
 totalScore += score;  
 }  
  
 public void displayStudentData() {  
 System.out.println("学号：" + studentId);  
 System.out.println("姓名：" + studentName);  
 System.out.println("成绩：" + score);  
 }  
  
 public static void displayStatistics() {  
 System.out.println("学生人数：" + studentCount);  
 System.out.println("平均成绩：" + (totalScore / studentCount));  
 }  
 public static void main(String[] args) {  
 // 创建学生对象并存储数据  
 StudentData student1 = new StudentData("001", "张三", 85.5);  
 StudentData student2 = new StudentData("002", "李四", 76.0);  
 StudentData student3 = new StudentData("003", "王五", 92.3);  
  
 student1.displayStudentData();  
 student2.displayStudentData();  
 student3.displayStudentData();  
  
 displayStatistics();  
 }  
}

1. 实现一个金丝猴类，要求其继承自动物类，并实现包括金丝猴是受保护而且是有价值的动物的多重继承的功能。

// 动物类  
class Animal {  
 void speak() {  
 System.out.println("动物发出声音");  
 }  
}  
// 金丝猴类，继承自动物类  
class GoldenMonkey extends Animal {  
 void speak() {  
 System.out.println("金丝猴发出声音");  
 }  
 void isProtected() {  
 System.out.println("金丝猴是受保护的动物");  
 }  
 void isValuable() {  
 System.out.println("金丝猴是有价值的动物");  
 }  
}  
// 测试类  
public class Main {  
 public static void main(String[] args) {  
 GoldenMonkey monkey = new GoldenMonkey();  
 monkey.speak(); // 调用金丝猴类的 speak 方法  
 monkey.isProtected(); // 调用金丝猴类的 isProtected 方法  
 monkey.isValuable(); // 调用金丝猴类的 isValuable 方法  
 }  
}

1. 实现如图所示的界面。



import java.awt.\*;

import javax.swing.\*;   
public class a05 {  
 public static void main(String[] args) {  
 JFrame frame = new JFrame();  
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   
 JPanel panel = new JPanel() {  
 @Override   
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 g.setColor(Color.BLUE);  
 g.fillRect(0, 0, 300, 200)；  
 g.setColor(Color.YELLOW);  
 g.fillRect(75, 50, 150, 100);  
 }  
 };  
 panel.setPreferredSize(new Dimension(300, 200));   
 frame.getContentPane().add(panel);   
 frame.pack();   
 frame.setVisible(true);   
 }  
}

1. 实现如图所示的界面，并实现将按钮“=”左侧两个文本框中的数相加的和显示在“=”右侧的文本框中的功能。



import javax.swing.\*;   
import java.awt.event.\*;   
public class a06 {  
 public static void main(String[] args) {  
 JFrame frame = new JFrame();  
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();   
 JTextField textField1 = new JTextField(10);

JLabel plusLabel = new JLabel("+");

JTextField textField2 = new JTextField(10);  
 JButton addButton = new JButton("=");

JTextField resultField = new JTextField(10);  
 resultField.setEditable(false);   
 addButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String text1 = textField1.getText();   
 String text2 = textField2.getText();  
 try {  
 int num1 = Integer.parseInt(text1);   
 int num2 = Integer.parseInt(text2);  
 int sum = num1 + num2;  
 resultField.setText(Integer.toString(sum));   
 } catch (NumberFormatException ex) {  
 resultField.setText("Invalid input");  
 }  
 }  
 });  
 panel.add(textField1);  
 panel.add(plusLabel);  
 panel.add(textField2);  
 panel.add(addButton);  
 panel.add(resultField);  
 frame.getContentPane().add(panel);  
 frame.pack();  
 frame.setVisible(true);  
 }  
}

7实现通过键盘输入一个浮点数，然后将该浮点数的整数部分输出的程序。

import java.util.Scanner;  
public class a08 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 System.out.print("请输入一个浮点数: ");  
 double number = scanner.nextDouble();  
  
 int integerPart = (int) number;   
  
 System.out.println("整数部分为: " + integerPart);  
  
 scanner.close();   
 }  
}

8计算并输出一个整数各位数字之和的程序，如1234的各位数字之和为1+2+3+4。

import java.util.Scanner;  
public class a09 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.print("请输入一个整数: ");  
 int number = scanner.nextInt();  
 int sum = 0;  
 int digit;  
 while (number != 0) {  
 digit = number % 10; // 取最后一位数字  
 sum += digit; // 累加到总和中  
 number /= 10; // 去掉最后一位数字  
 }  
 System.out.println("各位数字之和为: " + sum);  
 scanner.close();  
 }  
}

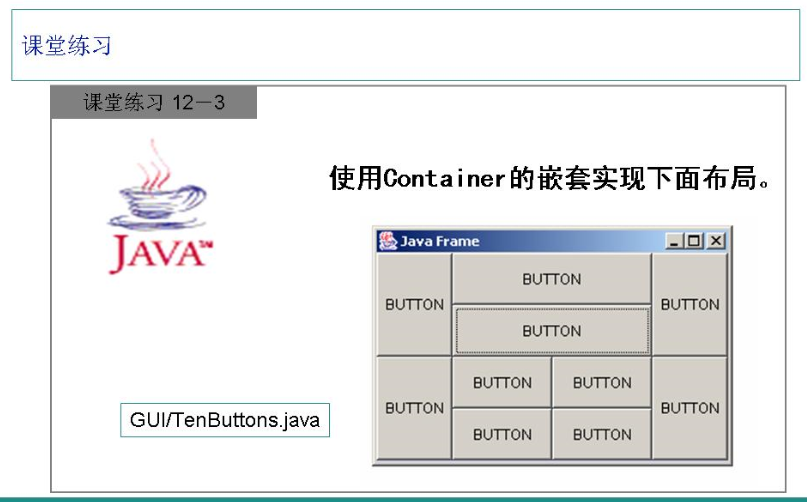
9实现字符串

String s= " SunJavaHPjavaOkjava兰理工java计通学院java计算机科学与技术java " 中统计“java”出现的次数的程序。

import java.util.Scanner;  
public class a09 {  
 public static void main(String[] args) {  
 String s = "SunJavaHPjavaOkjava 兰理工 java 计通学院 java 计算机科学与技术 java";  
 String target = "java";  
 int count = countSubstring(s, target);  
 System.out.println( "\""+target + "\" 出现的次数为: " + count);  
 }  
 public static int countSubstring(String s, String target) {  
 int count = 0;  
 int index = 0;  
 while (index != -1) {  
 index = s.indexOf(target, index); // 在字符串 s 中查找子串 target  
 if (index != -1) {  
 count++;  
 index += target.length(); // 继续从下一个位置开始查找  
 }  
 }  
 return count;  
 }  
}

10以m行n列的二维数组为参数进行方法调用，分别计算二维数组各列元素之和，返回并输出所计算的结果。

11实现如图所示的界面。



import java.awt.\*;  
  
public class Demo11 {  
 public static void main(String[] args) {  
 Frame f = new Frame("JavaFrame");  
  
 f.setLayout(new GridLayout(2,1));  
 f.setLocation(300,400);  
 f.setSize(300,200);  
  
 f.setBackground(new Color(204, 204, 255));  
 Panel p1 = new Panel(new BorderLayout());  
 Panel p2 = new Panel(new BorderLayout());  
 Panel p11 = new Panel(new GridLayout(2,1));  
 Panel p21 = new Panel(new GridLayout(2,2));  
  
 p1.add(new Button("Button"),BorderLayout.WEST);  
 p1.add(new Button("Button"),BorderLayout.EAST);  
  
 p11.add(new Button("Button"));  
 p11.add(new Button("Button"));  
 p1.add(p11,BorderLayout.CENTER);  
  
 p2.add(new Button("Button"),BorderLayout.WEST);  
 p2.add(new Button("Button"),BorderLayout.EAST);  
  
 for (int i = 0; i < 4; i++) {  
 p21.add(new Button("Button"));  
 }  
 p2.add(p21,BorderLayout.CENTER);  
 f.add(p1);  
 f.add(p2);  
 f.setVisible(true);  
 }  
}

12、 15个红球和15个绿球排成一圈，从第1个球开始数，当数到第13个球时就拿出此球，然后再从下一个球开始数，当再数到第13个球时又取出此球，如此循环进行，直到仅剩15个球为止，问怎么样排法才能每次取出的球都是红球。

public class GetRedBalls5\_6 {

public static void main(String[] args){

int[] ball=new int[30];

int i, j, t;

for(i=0; i<30; i++)

ball[i]=i+1;

System.out.println("取出红球的编号为: ");

for(i=0, j=0, t=30; t!=15; i++) {

if(i==30)

i=0;

if(ball[i]!=0)

j++;

if(j==13&&ball[i]!=0) {

ball[i]=0;

System.out.print(" "+(i+1)+" ");

j=0;

t--;

}

}

}

13、请按照以下要求设计一个学术类Student和它的一个子类Undergraduate ，并进行测试。要求如下：

（1）Student类中包含姓名（字符串类型）、年龄（整型）两个属性。

（2）Student类中定义两个构造方法。一个是无参构造方法；一个接收两个参数的构造方法，两个参数分别为姓名和年龄属性赋值。

（3）Student包括以下成员方法：

void setName(String name); 设置姓名

void setAge(float grade)； 设置年龄

String getName(); 获取姓名

float getAge(); 获取年龄

Void show(): 输出姓名和年龄

（4）Undergraduate类继承Student类，并增加一个degree（学位）属性。有一个包括三个参数的构造方法，前两个参数用于给继承的name和age属性赋值，第三个参数给degree专业赋值，一个show（）方法用于打印Undergraduate的三个属性信息。

（5）在测试类中分别创建Student对象和Undergraduate对象，调用它们的show().方法。

class Student{  
 private String name;  
 private int age;  
 public Student(){  
  
 }  
 public Student(String name,int age){  
 this.name=name;  
 this.age=age;  
 }  
 public void setName(String name){  
 this.name=name;  
 }  
 public void setAge(int age){ //  
 this.age=age;  
 }  
 public String getName(){  
 return name;  
 }  
 public int getAge(){  
 return age;  
 }  
 public void show(){  
 System.out.println("学生姓名"+name);  
 System.out.println("学生年龄"+age);  
 }  
}  
class Undergraduate extends Student{  
 private String degree;  
 public Undergraduate(String name,int age,String degree){  
 super(name,age);  
 this.degree=degree;  
 }  
 public void show(){  
 super.show();  
 System.out.println("学生学位"+degree);  
 }  
}  
public class test12 {  
 public static void main(String[] args){  
 Student stu1=new Student("zhangsan",25);  
 stu1.show();  
 System.out.println();  
 Undergraduate stu2=new Undergraduate("lisi",26,"buoshi");  
 stu2.show();  
 }  
}

14 设有一个动物声音模拟器，希望模拟器可以模拟许多动物的叫声，要求如下：

（1）编写接口Animal。Animal接口有两个抽象方法cry()和getAnimalName()，要求实现该接口的各种具体动物类给出自己的叫声和种类名称。

（2）编写模拟器类Simulator。该类有一个playSound(Animal animal)方法，该方法的参数是Animal类型，即参数animal可以调用实现Animal接口类重写的cry()方法播放具体动物的声音，调用重写的getAnimalName()方法显示动物种类的名称。

（3）编写实现Animal接口的Dog类和Cat类。

（4）编写主类MainClass,在主类的main方法中至少包含如下代码：

Simulator simulator=new Simulator()

simulator.playSound(new dog())

simulator.playSound(new cat())

interface Animal {  
 void cry();  
 String getAnimalName();  
}  
// Dog 类实现 Animal 接口  
class Dog implements Animal {  
 public void cry() {  
 System.out.println("汪汪汪！");  
 }  
 public String getAnimalName() {  
 return "狗";  
 }  
}  
// Cat 类实现 Animal 接口  
class Cat implements Animal {  
 public void cry() {  
 System.out.println("喵喵喵！");  
 }  
 public String getAnimalName() {  
 return "猫";  
 }  
}  
// 模拟器类  
class Simulator {  
 public void playSound(Animal animal) {  
 animal.cry();  
 System.out.println("动物种类：" + animal.getAnimalName());  
 }  
}  
public class test13 {  
 public static void main(String[] args) {  
 Simulator simulator = new Simulator();  
 simulator.playSound(new Dog());  
 simulator.playSound(new Cat());  
 }  
}

15、使用Java的输入和输出流将一个文本文件的内容按行读出，每读出一行就顺序添加行号，并写入另一个文件中。（读取的文件路径：E:/Source.txt；写入的文件路径：E:/Object.txt）

import java.io.\*;  
public class test13 {  
 public static void main(String[] args) {  
 String inputFile = "E:/Source.txt";  
 String outFile = "E:/Object.txt";  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(inputFile));  
 BufferedWriter writer = new BufferedWriter(new FileWriter(outFile))) {  
 String line;  
 int lineNumber = 1;  
 while ((line = reader.readLine()) != null) {  
 String lineWithLineNumber = lineNumber + ": " + line;  
 writer.write(lineWithLineNumber);  
 writer.newLine();  
 lineNumber++;  
 }  
 System.out.println("File line numbering completed.");  
 } catch (IOException e) {  
 System.out.println("An error occurred: " + e.getMessage());  
 }  
 }  
}

16、编写一个图形用户界面的应用程序，包括两个文本框和一个按钮，当单击按钮时，可以把一个文本框中的内容复制到另一个文本框中。

import java.awt.\*;  
import javax.swing.\*;  
import java.awt.event.\*;  
public class test16 {  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("文本框复制");  
 frame.setLayout(new FlowLayout());  
 frame.setSize(300, 150);  
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 JTextField textField1 = new JTextField(20);  
 JTextField textField2 = new JTextField(20);  
 JButton button = new JButton("复制");  
 button.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 String text = textField1.getText();  
 textField2.setText(text);  
 }  
 });  
 frame.add(textField1);  
 frame.add(button);  
 frame.add(textField2);  
 frame.setVisible(true);  
 }  
}

17、请编写一个类，类名为MulThread ，该类实现Runnable接口。类中定义了含一个字符串参数的构造函数，设置线程名称；其中的run( )方法如下实现：方法中先在命令行显示“begin”+线程名，然后休眠1秒的时间，最后显示线程结束信息: "finished"+线程名。编写Application，在其中通过Runnable创建MulThread类的三个线程对象t1、t2、t3，并启动这三个线程。

class MulThread implements Runnable {  
 private String threadName;  
  
 public MulThread(String threadName) {  
 this.threadName = threadName;  
 }

public void run() {  
 System.out.println("begin " + threadName);  
 try {  
 Thread.sleep(1000); // 休眠1秒  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.out.println("finished " + threadName);  
 }  
}  
public class test17 {  
 public static void main(String[] args) {  
 MulThread t1 = new MulThread("Thread 1");  
 MulThread t2 = new MulThread("Thread 2");  
 MulThread t3 = new MulThread("Thread 3");  
  
 Thread thread1 = new Thread(t1);  
 Thread thread2 = new Thread(t2);  
 Thread thread3 = new Thread(t3);  
  
 thread1.start();  
 thread2.start();  
 thread3.start();  
 }  
}

18、编程实现：

（1）编写一个圆类Circle，该类拥有：

①一个成员变量Radius（私有，浮点型）； // 存放圆的半径；

②两个构造方法Circle( ) // 将半径设为0

Circle(double r ) //创建Circle对象时将半径初始化为r

③ 三个成员方法

double getArea( ) //获取圆的面积

double getPerimeter( ) //获取圆的周长

void show( ) //将圆的半径、周长、面积输出到屏幕

（2）编写一个圆柱体类Cylinder，它继承于上面的Circle类。还拥有：

①一个成员变量double hight（私有，浮点型）； // 圆柱体的高；

②构造方法 Cylinder (double r, double h ) //创建Circle对象时将半径初始化为r

③ 成员方法double getVolume( ) //获取圆柱体的体积

void showVolume( ) //将圆柱体的体积输出到屏幕

（3）编写应用程序，创建类的对象，分别设置圆的半径、圆柱体的高，计算并分别显示圆半径、圆面积、圆周长，圆柱体的体积。

class Circle{  
 private double Radius;  
 public Circle() {  
 Radius = 0;  
 }  
 public Circle(double r){  
 Radius=r;  
 }  
 public double getArea( ){  
 return Math.PI \* Radius \* Radius;  
 }  
 public double getPerimeter( ){  
 return 2 \* Math.PI \* Radius;  
 }  
 public void show( ){  
 System.out.println("半径：" + Radius);  
 System.out.println("周长：" + getPerimeter());  
 System.out.println("面积：" + getArea());  
 }  
}  
class Cylinder extends Circle {  
 private double height;  
 public Cylinder(double r, double h) {  
 super(r);  
 height = h;  
 }  
 public double getVolume() {  
 return getArea() \* height;  
 }  
 public void showVolume() {  
 System.out.println("体积：" + getVolume());  
 }  
}  
public class test18 {  
 public static void main(String[] args) {  
 Circle circle = new Circle(5);  
 circle.show();  
 System.out.println();  
 Cylinder cylinder = new Cylinder(3, 8);  
 cylinder.showVolume();  
 }  
}

19、按要求编写一个Java应用程序程序：

（1）定义一个接口CanFly，描述会飞的方法public void fly();

（2）分别定义类飞机和鸟，实现CanFly接口。

（3）定义一个测试类，测试飞机和鸟，在main方法中创建飞机对象和鸟对象，

再定义一个makeFly()方法，其中让会飞的事物飞。并在main方法中调用该方法，

让飞机和鸟起飞。

interface CanFly{  
 void fly();  
}  
class Bird implements CanFly{  
 @Override  
 public void fly() {  
 System.out.println("鸟儿飞翔！");  
 }  
}  
class Plan implements CanFly{  
 @Override  
 public void fly() {  
 System.out.println("飞机起飞！");  
 }  
}  
public class test19 {  
 public static void main(String[] args){  
 Plan p=new Plan();  
 Bird b=new Bird();  
 makeFly(p);  
 makeFly(b);  
 }  
 public static void makeFly(CanFly flyingObject){  
 flyingObject.fly();  
 }  
}

20、应用FileInputStream类，编写应用程序，从磁盘上读取一个Java程序，并将源程序代码显示在屏幕上。（被读取的文件路径为：E:/Hello.java）

import java.io.FileInputStream;

import java.io.IOException;

public class ReadJavaFile {

public static void main(String[] args) {

// 指定文件路径

String filePath = "E:/Hello.java";

// 创建文件输入流

try (FileInputStream fis = new FileInputStream(filePath)) {

// 创建一个字节数组来保存文件内容

byte[] content = new byte[(int) fis.available()];

// 读取文件内容到字节数组

fis.read(content);

// 将字节数组转换为字符串并打印在屏幕上

String sourceCode = new String(content);

System.out.println(sourceCode);

} catch (IOException e) {

e.printStackTrace();

}

}

}

21、编写程序，在屏幕上显示带标题的窗口，并添加一个按钮。当用户单击按钮时，结束程序。

import javax.swing.JButton;

import javax.swing.JFrame;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class WindowWithButton {

public static void main(String[] args) {

// 创建窗口

JFrame frame = new JFrame("带按钮的窗口");

// 设置窗口大小

frame.setSize(300, 200);

// 创建按钮

JButton button = new JButton("结束程序");

// 添加按钮点击事件监听器

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// 点击按钮时，退出程序

System.exit(0);

}

});

// 将按钮添加到窗口

frame.getContentPane().add(button);

// 设置窗口可见

frame.setVisible(true);

}

}

import javax.swing.\*；public class test21{

public static void main（String args）{

JFrame frame = new JFrame（"带标题的窗口"）；frame.setSize（300，200）；frame.setDefaultCloseOperation（JFrame.EXIT\_ON\_CLOSE）；JButton button = new JButton（"结束程序"）；frame.getContentPane（）.add（button）

button.addActionListener（e->{

JOptionPane.showMessageDialog（null，"程序已结束"）；frame.dispose（）；

}）；

frame.setVisible（true）；

}

}

22、编写一个JDBC程序，要求如下：

（1） 驱动程序：com.mysql.jdbc.Driver

url：jdbc:mysql://localhost:3306/mydb?characterEncoding=utf8

用户名：root 密码：root

表信息：student(学号，姓名，年龄，成绩)

（2）向student表中插入一条记录（1，孙悟空，500，89.8）

（3）查询并显示student表中的所有记录。

import java.sql.\*;

public class JDBCExample {

public static void main(String[] args) {

// JDBC连接信息

String url = "jdbc:mysql://localhost:3306/mydb?characterEncoding=utf8";

String username = "root";

String password = "root";

// SQL语句

String insertQuery = "INSERT INTO student (学号, 姓名, 年龄, 成绩) VALUES (1, '孙悟空', 500, 89.8)";

String selectQuery = "SELECT \* FROM student";

try {

// 加载驱动程序

Class.forName("com.mysql.jdbc.Driver");

// 建立数据库连接

Connection connection = DriverManager.getConnection(url, username, password);

// 创建Statement对象

Statement statement = connection.createStatement();

// 插入记录

statement.executeUpdate(insertQuery);

System.out.println("成功插入一条记录。");

// 查询记录

ResultSet resultSet = statement.executeQuery(selectQuery);

System.out.println("查询结果：");

while (resultSet.next()) {

int 学号 = resultSet.getInt("学号");

String 姓名 = resultSet.getString("姓名");

int 年龄 = resultSet.getInt("年龄");

double 成绩 = resultSet.getDouble("成绩");

System.out.println("学号: " + 学号 + ", 姓名: " + 姓名 + ", 年龄: " + 年龄 + ", 成绩: " + 成绩);

}

// 关闭连接

resultSet.close();

statement.close();

connection.close();

} catch (ClassNotFoundException e) {

System.out.println("无法加载驱动程序。");

e.printStackTrace();

} catch (SQLException e) {

System.out.println("数据库操作出现异常。");

e.printStackTrace();

}

}

}

JAVA语言基础

**1.从键盘输入一个浮点数，将该浮点数的整数部分和小数部分分别输出。**

import java.util.\*;

public class Test4\_1 {

public static void main(String[ ] args) {

Scanner reader = new Scanner(System.in);

System.out.print("请输入一个浮点数：");

double num = reader.nextDouble( );

int digital = (int) num;

System.out.println(num+"的整数部分为"+digital);

System.out.println(num+"的小数部分为"+(num-digital));

}

}

1. **编程实现：**

（1）定义一个人类Person，包含的内容如下：成员变量----姓名name、年龄age；成员方法----设置成员变量的值setNameAge( )、显示成员变量的值show( )；

（2） 定义Person类的子类学生类Student，包含的内容如下：成员变量----系别department；成员方法----设置成员变量的值setDepartment( )；

（3）在主方法main( )中创建学生类Student的对象，并调用方法setNameAge( )，show( )和setDepartment( )显示相应的信息。（12分）

class Person {

private String name;

private int age;

public void setNameAge(String name, int age) {

this.name=name;

this.age=age;

}

public void show() {

System.out.println("姓名："+name+" 年龄："+age);

}

}

class Student extends Person{

private String department;

public void setDepartment(String dep) {

department=dep;

System.out.println("我是"+department +"的学生");

}

}

public class Test4\_3{

public static void main(String[] args) {

Student stu = new Student ();

stu.setNameAge("张三",20);

stu.show();

stu.setDepartment("数学系");

}

}

1. **//filename:App8\_11.java  接口的实现**  
   interface IShape  
   {  
   final double PI=3.14;  
   abstract double getArea();  
   abstract double getLength();  
   }  
   class Circle implements IShape  
   {  
   double radius;  
   public Circle(double r)  
   {  
   radius=r;  
   }  
   public double getArea()  
   {  
   return PI\*radius\*radius;  
   }  
   public double getLength()  
   {  
   return 2\*PI\*radius;  
   }  
   }  
   class Rectangle implements IShape  
   {  
   private double width;  
   private double height;  
   public Rectangle(double width,double height)  
   {  
   this.width=width;  
   this.height=height;  
   }  
   public double getArea()  
   {  
   return 2\*(width+height);  
   }  
   public double getLength()  
   {  
   return 2\*(width+height);  
   }  
   }  
   public class App8\_11  
   {  
   public static void main(String[] args)  
   {  
   IShape circle=new Circle (5.0);  
   System.out.print("圆面积="+circle.getArea());  
   System.out.println(";周长="+circle.getLength());  
   Rectangle rect=new Rectangle(6.5,10.8);  
   System.out.print("矩形面积="+rect.getArea());  
   System.out.print(";周长="+rect.getLength());  
   }  
   }
2. **抽象类的应用举例，定义一个形状抽象类Shape以该抽象类为父类派生出圆形子类Circle和矩形子类Rectangle**  
   //filename: App8\_ 10. java 抽象类的说明  
   abstract class Shape //定义形状抽象类Shape  
   ｛  
          protected String name;  
          public Shape( String xm) //抽象类中的一般方法,本方法是构造方法  
   ｛  
          name=xm;  
         System. out. print("名称:"+ name) ;  
   ｝  
   abstract public double getArea();  
   abstract public double getLength(); //声明抽象方法  
   ｝  
   class Circle extends Shape //定义继承自Shape的圆形子类Circle  
   {  
         privatefinal double PI= 3.14;  
          private double radius;  
          public Circle(String shapeName,，double r) //构造方法  
   ｛  
           super( shapeName);  
           radius = r;  
   ｝  
   public double getArea() //实现抽象类中的getArea()方法  
   ｛  
            return PI \* radius￥radius;  
   ｝  
   public double getLength() //实现抽象类中的getLength()方法  
     ｛  
   return 2 \* PI \* radius;  
      ｝  
   ｝  
   class Rectangle extends shape //定义继承自Shape 的矩形子类Rectangle  
   ｛  
   private double width;

private double height;  
public Retangle(String shapeName,double width, double height) //构造方法  
｛  
      super( shapeName);  
      this. width= width;  
      this. height = height;  
｝  
public double getArea( ) //实现抽象类中的getArea(()方法  
｛  
      return width＊height;  
｝  
public double getLength( ) //实现抽象类中的getLength()方法  
｛  
      return 2 \* (width + height) ;  
｝  
｝  
public class App8\_ 10 //定义主类  
｛  
public static void main( String[] args)  
｛  
       Shape rect = new Rectangle("长方形",6.5,10.3); //声明父类对象，指向子类对象  
System. out. print(";面积="+ rect. getArea());  
System. out. println(";周长="+rect. getLength());  
Shape circle= new Circle("圆"，10.2);//声明父类对象circle,指向子类对象  
System. out. print(";面积="+ circle. getArea());  
System. out. println(";周长="+ circle. getLength());  
｝  
｝  
程序运行结果为:  
名称:长方形;面积=66.95;周长=33.6  
名称:圆;面积= 326, 65599999997;周长=64.056

**5. 从键盘上输入一个月份，然后判断该月份的天数。**

//filename:App4\_4.java

import java.util.\*;

public class App4\_4

{

public static void main(String[]args)

｛

int month,days;

Scanner reader= new Scanner(System.in);

System.out.print("请输人月份:");

month=reader.nextInt();

switch(month)

｛

case 2: days=28;

break;

case 4:

case 6:

case 9:

case 11: days=30;

break;

default:days=31;

}

System.out.println(month+"月份为"+days+"天");

}

程序运行结果为:请输入月份:6

6月份为30天

**6.设数组中有n个互不相同的数,不用排序求出其中的最大值和次最大值。**

//filename: App5\_2.java

public class App5\_2

{

public static void main(String[] args)

｛

int i,max, sec;

int[]a={8,50,20,7,81,55,76,93};

if(a[0]>a[1])9

{

max= a[0];

sec=a[1];

}

else

｛

max=a[1];

sec=a[0];

｝

System.out.print("数组的各元素为:"+a[0]+" "+a[1]);

for(i=2;i<a.length;i++)

｛

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

if(a[i]>max)

｛

sec=max;

max=a[i];

｝

else

if(a[i]>sec) sec＝a[i];

｝

System.out.print(”\n其中的最大值是:"+max);

System.out.println(" 次最大值是:"+sec);

}

}

程序运行结果为:

数组的各元素为:85020781 557693其中的最大值是:93 次最大值是:81

**7.在圆柱体类Cylinder中,使用构造方法的重载。**

//filename: App7\_5.java

class Cylinder

{

private double radius;

private int height;

private double pi = 3.14;

String color;

public Cylinder()

｛

radius=1;

height=2;

color="绿色";

｝

public Cylinder(doublerinthStringstr)

｛

radius\*r;

height=h;

color=stri ;

｝

public void setColor()

｛

System.out,println("该圆柱的颜色为:"+color);

｝

double area()

｛

return pi \* radius \* radius;

}

double volume()

｛

return area() \* height;

}

}

public class App7\_5

{

public static void main(String[] args)

{

Cylinder volul=new Cylinder();

System.out.println("圆柱1底面积="+ volu1.area());

System.out.println("圆柱1体积="+volu1.volume());

volul.setColor();

Cylinder volu2= new Cylinder(2.5,8,"红色");

System.outprintln("圆柱2底面积="+ volu2.area());

System.out.println("圆柱2体积="+volu2.volume());

volu2.setColor();

}

}

**8.以Person作为父类,创建学生子类Student,并在子类中调用父类里某指定的构造方法。**

调用父类中的特定构造方法

//filename: App8 2.java

class Person｛

private String name;

private int age;

public Person()

｛

System.out.printIn("调用了 Person 类的无参构造方法");

｝

public Person(String name, int age)

｛

System.out.println("调用了Person类的有参构造方法");

this.name=name;

this.age=age;

｝

public void show()

｛

System.out.println("姓名:"+name+" 年龄:"+age);

}

class Student extends Person

{

private String department;

public Student()

｛

System.out.println("调用了学生类的无参构造方法Student()");

｝

public Student(String name,int ageString dep)

｛

super(name,age);

department =dep;

System.out.println("我是"+department+"的学生");

System.out.println("调用了学生类的有参构造方法 Student(String name,int age,String dep)")

}

}

public class App8\_2

{

public static void main(String[] args)

{

Student stu1 = new Student();

Student stu2=new Student("李小四",23，"信息系");

stul.show();

stu2.show();

}

}

1. **利用圆柱体类Cylinder来介绍静态方法的使用。**

//filename:App7\_9.java

class Cylinder｛

private static int num=0;

private static double pi=3.14;private double radius;

private int height;

public Cylinder(double r,int h)｛

radius=r;height=h;num++； ｝//当构造方法Cylinder()被调用时，num便加1

public static void count() ｛//声明 count()为静态方法

System.out.println(”创建了”+num+”个对象”)；

)｝

double area( )｛

return pi radius \* radius;｝

double volume()

｛

return area( ) \* height;｝

public class App7\_9 //主类

public static void main(String[] args)

｛

Cylinder.count(); /在对象产生之前用类名Cylinder调用count()方法Cylinder volul =new. Cylinder(2.5,3);

volul.count(); //用对象 volu1 调用 count()方法

System.out.println(”圆柱1的体="+volu1.volume());

Cylinder volu2=new. Cylinder(1.0,2);

Cylinder.count(); //用类名Cylinder直接调用count()方法System.out.println(”圆柱2的体积="+ volu2.volume());