## 第1章

一、问答题

1．Java有哪些特点？

Java是一种可以撰写跨平台应用程序的面向对象的程序设计语言。Java 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于PC、数据中心、游戏控制台、科学超级计算机、移动电话和互联网，同时拥有全球最大的开发者专业社群。

Java语言是简单的。Java语言的语法与C语言和C++语言很接近，使得大多数程序员很容易学习和使用Java。Java语言不使用指针，而是引用。并提供了垃圾回收机制，使得程序员不必为内存管理而担忧。

Java语言是纯粹面向对象的。Java语言提供类、接口和继承等。

Java语言是分布式的。Java语言支持Internet应用的开发，Java的RMI（远程方法激活）机制也是开发分布式应用的重要手段。

Java语言是健壮的。Java的强类型机制、异常处理、垃圾的自动收集等是Java程序健壮性的重要保证。安全检查机制使得Java更具健壮性。

Java语言是安全的。Java提供了一个安全机制以防恶意代码的攻击。Java对通过网络下载的类具有一个安全防范机制，并提供安全管理机制让Java应用设置安全哨兵。

Java语言是体系结构中立的。Java程序（后缀为java的文件）在Java平台上被编译为体系结构中立的字节码格式（后缀为class的文件），然后可以在实现这个Java平台的任何系统中运行。这种途径适合于异构的网络环境和软件的分发。

Java语言是跨平台的。Java程序的字节码文件可以在任何一个操作系统上运行。

2．编译Java Application程序的命令是什么？运行命令是什么？

编译命令：javac 源程序的文件名

运行命令：java main方法所在类的类名

3．Java应用程序分几类？各有什么特点？

不同的Java程序运行在不同的环境中，习惯上将运行环境相同的Java程序归为一类，可以分3类：

Java Application应用程序，属于JavaSE

Applet小程序（目前实际开发中基本不使用）

Servlet程序（在JavaEE中使用）

二、编程题

编写一个文件名为FirstProgram的Java Application，功能是输出“Hello,EveryOne!”，写出编译和运行文件的命令和编译器产生的文件名。

程序源码：FirstProgram.java

public class FirstProgram{

public static void main(String args[]){

System.out.println(“Hello,EveryOne!”);

}

}

编译命令：

javac FirstProgram.java

运行命令：

java FirstProgram

## 第2章

1．编程实现键盘输入一个人的出生年份，计算出他这辈子能过几个闰年。（以100岁为寿命长度。）

package chapter2;

import java.util.Scanner;

/\*

\*编程实现键盘输入一个人的出生年份，计算出他这辈子能过几个闰年。（以100岁为寿命长度。）

\*/

public class GetYear {

public static void main(String args[]) {

System.out.println("请输入出生年份");

Scanner reader = new Scanner(System.in);

int firstYear = reader.nextInt();

int length = 100;

int year;

for (int i = 0; i <= length; i++) {//包含第100年

year = firstYear + i;

if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)

System.out.println(year);

}

}

}

2．用while循环求Fibonacci序列。

1， 1， 2， 3， 5， 8， 13， 21， 34， 55， 89……

package excercise.ch2;

/\*\*

\* 斐波那契数列（Fibonacci sequence），又称黄金分割数列.

\* 因数学家列昂纳多·斐波那契（Leonardoda Fibonacci）以兔子繁殖为例子而引入，

\* 故又称为“兔子数列”，指的是这样一个数列：1、1、2、3、5、8、13、21、34、……在数学上，

\* 斐波纳契数列以如下被以递推的方法定义：F(1)=1，F(2)=1, F(n)=F(n-1)+F(n-2)（n>=3，n∈N\*）

\* @author Administrator

\*

\*/

class GetFibonacci {

//getFib(int n)用于计算斐波那契数列中第n项的值

static int getFib(int n) {

if(n==0||n==1)

return 1;

return getFib(n-2)+getFib(n-1);

}

public static void main(String args[]) {

int max=20;//求0-max项范围内的斐波那契数列

int i=0;

while(i<20){

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

i++;

if(i%5==0)

System.out.println("");//每5个数以后换行

}

}

}

3．用辗转相除法求两个整数的最大公因数。

package chapter2;

/\*\*

\* 辗转相除法， 又名欧几里德算法（Euclidean algorithm），是求最大公约数的一种方法。

\* 它的具体做法是：用较大数除以较小数，再用出现的余数（第一余数）去除除数，

\* 再用出现的余数（第二余数）去除第一余数，如此反复，直到最后余数是0为止。

\* 如果是求两个数的最大公约数，那么最后的除数就是这两个数的最大公约数。

\* @author Administrator

\*

\*/

public class GreaterCommonFactor {

//求a、b的最大公约数的方法

static void getGreaterCommonFactor(int a,int b) {

int k = 0;

System.out.print("gcd(" + a + "，" + b + ")=");

do {

k = a % b;//如果a<b,则余数k=a,实际上下次循环a、b进行了交换

a = b;//让a等于除数

b = k;

} while (k != 0);//当余数k=0时的除数a就是最大公约数

System.out.println(a);

}

public static void main(String args[]) {

getGreaterCommonFactor(12,18);

}

}

4．统计一个字符数组中每个字母出现的次数。

package excercise.ch2;

public class CounteLettersInArray2 {

public static void main(String[] args) {

//letters[]是被统计数组

char[] letters={'a','b','a','c','c','m','s','a','t','b','c'};

//count[]保存26个字母出现的次数,count[0]保存a的次数，count[25]保存z的次数

int[] count=new int[26];

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

count[letters[i]-'a']++; //letters[i]-'a'如果是0，表示a，如果是1表示b

}

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

if(count[i]>0)

System.out.println((char)(i+'a')+":"+count[i]);

}

}

}

5．找到数组中所有的数组元素大于平均值的那些元素。

package chapter2;

import java.util.Scanner;

public class AnalyzeNumber {

/\*\*

\* 数组中所有的数组元素大于平均值的那些元素

\*/

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.println("请输入要存储的个数");

int n = input.nextInt();

double[] numbers = new double[n];

double sum = 0;//保存数组的元素之和

System.out.println("请输入您想存储的数字");

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

numbers[i] = input.nextDouble();

sum += numbers[i];

}

double avg = sum / n;

System.out.println("平均值为：" + avg);

System.out.println("大于平均值" + avg+"的元素有：");

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

if (numbers[i] > avg) {

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

}

}

}

}

6．随机产生20个[77,459)的正整数存放到数组中，求数组中的最大值、最小值、平均值及各个元素之和。

package chapter2;

/\*

\*随机产生20个[77,459)的正整数存放到数组中，求数组中的最大值、最小值、平均值及各个元素之和。

\*/

public class ArrayDemo {

static void getArray() {

// 声明一个长度为20的数组

int[] array = new int[20];

// 遍历数组

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

//Math.random()会产生[0,1)区间中的双精度浮点随机数

int num = (int) (Math.random() \* (459 - 77) + 77);//产生[77,459)之间的整数

array[i] = num;

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

}

// 声明最大值，最小值，和，平均值

int max = 0;

int min = array[0];

int sum = 0;

int average = 0;

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

// 最大值

if (max < array[i]) {

max = array[i];

}

// 最小值

if (min > array[i]) {

min = array[i];

}

// 和值

sum = sum + array[i];

}

// 平均值

average = sum / array.length;

// 打印

System.out.println("\n最大值是" + max);

System.out.println("最小值是" + min);

System.out.println("和是" + sum);

System.out.println("平均值是" + average);

}

public static void main(String args[]) {

getArray();

}

}

## 第3章

一、问答题

1．面向对象的四个基本特征是什么？如何理解他们？

答：面向对象程序设计包括四个基本特征：抽象、封装、继承和多态。

抽象是根据业务需要，把客观世界的事物中与业务相关的特征和功能归纳总结出来、忽略与本业务无关的特征和功能，从形成类的过程。

封装是指把同一类事物的特征和功能都定义到一个类里，变成类的属性和方法。同时通过访问控制符来定义每个属性和方法的可见性。

继承是类之间的一种重用机制。如果A类继承了B类，则A类拥有了B类的除了私有属性和私有方法外的所有属性和方法，同时可以添加自己特有的属性和方法。

多态是实现接口的多个类或一个父类的多个子类有相同的方法名，但是具有不同的表现方式。例如动物类具有喊叫的方法，但是它的子类猫和子类狗的喊叫效果却不一样。

2．java源程序的结构是什么？

答：源程序结构如下：

package 包名;

import 包名.类名;

import 包名.类名;

…

[public ] class 类名{

属性变量定义… …

方法定义… … }

… …

[public ] class 类名{属性变量定义… …

方法定义… … }

程序结构说明如下：

（1）package语句放在第一行，且只能有一个package语句；

（2）如果类中使用到了其他包中的类，可以写import语句。import语句必须放在package语句之后、类的定义之前。如果引入多个包中的类，可以写多行import语句。可以用“包名.\*”表示引入该包名里的所有类，但不包括子包下的类；

（3）类的定义要放在import语句的后面。一个java源文件可以包括多个类的定义，但是只有一个类可以定义成public。如果另一个类也需要定义成public则必须单独再建一个java源程序文件。

3．成员变量应该放在什么位置定义？方法内定义的变量是成员变量吗？

成员变量应该放在类的里面并且在方法的外面。方法内定义的变量不是成员变量，只是局部变量。

4．如何创建对象？this代表什么？

对象的创建也称为对象的实例化，是在内存开辟一个区域，保存新创建的对象，一般格式为： 对象名=new 构造方法（参数列表）;

new关键字表示创建对象的意思，构造方法是类中定义的构造方法，参数列表是可选的，根据实际的构造方法的参数来写。

this代表当前对象，也就是当前方法所在的对象。一个类可以创建多个对象，而且实例方法肯定是属于一个对象的，但是在类体内还不知道将来创建的对象是什么，所以就用this来代表该方法所在的对象。

5．包的作用是什么？带包类如何执行？环境变量classpath如何配置？

通过包的定义可以起到两个作用：一是解决命名冲突问题，在同一个包中的类不能重名，但在不同包中的类可以重名；二是通过包以及访问权限控制符可以限制不同包之间类的访问权限。

运行时要用java 报名.类名。

Classpath也该设置为该包所在的文件夹，而不是类文件本身所在的文件夹。

二、编程题

编写一个三角形类，要求能够计算三角形的面积和周长。从键盘输入一个三角形的3条边的长度，并用这三条边创建一个三角形对象，输出此三角形对象的面积和周长。

package excercise.ch3;

import java.util.Scanner;

public class Triangle {

private double a,b,c;

public double getA() {

return a;

}

public void setA(double a) {

this.a = a;

}

public double getB() {

return b;

}

public void setB(double b) {

this.b = b;

}

public double getC() {

return c;

}

public void setC(double c) {

this.c = c;

}

public Triangle(double a, double b, double c) {

super();

this.a = a;

this.b = b;

this.c = c;

}

public double getArea(){

double p=(a+b+c)/2;

return Math.sqrt(p\*(p-a)\*(p-b)\*(p-c));

}

public double getPerimeter(){

return a+b+c;

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

double a=sc.nextDouble();

double b=sc.nextDouble();

double c=sc.nextDouble();

if(!(a+b>c&&a+c>b&&b+c>a)){

System.out.println("您输入的3个边长不能构成一个三角形");

return;

}

Triangle triangle=new Triangle(a,b,c);

System.out.println("面积："+triangle.getArea());

System.out.println("周长："+triangle.getPerimeter());

}

}

## 第4章 继承

1.this有哪几种用法？super有哪几种用法？

this主要有三种用法。

（1）用this来引用当前对象的成员变量或成员方法。

（2）用this来调用类中的另一个构造方法。

（3）把this当做参数传递给其他方法

super主要有两种用法。

（1）引用父类的成员变量或成员方法。

1. 调用父类的构造方法。

2.变量的隐藏和方法的覆盖有什么区别？

当父类和子类的属性名字一样时，在子类中默认使用的是子类定义的属性，除非明确使用super来调用，所以在子类对象的内存里实际上有2个同名的变量，默认使用的是子类的变量，这种现象称作子类变量隐藏了父类的同名变量。

当父类和子类定义的方法名一样时，子类对象内存里只有子类定义的方法，不再有父类定义的方法，这种现象称作子类的方法覆盖了父类的同名方法。

3.请介绍一下4种访问控制符的作用。

java一共定义了四种访问权限控制符：private、默认权限、protected、public，通过权限控制符和包能够控制访问权限。

（1）private

private可以用来修饰成员变量、成员方法、构造方法以及内部类（内部类是定义在另一个类里面的类，跟成员变量一样属于另一个类的成员）。private修饰的成员只能被它所属的类内的其他方法访问，不能被该类以外的其他类访问。

（2）默认访问权限

默认访问权限也称为包访问权限，不写访问控制符的就是默认访问权限。默认访问权限可以用来修饰成员变量、成员方法、构造方法、类和接口。默认访问权限定义的内容可以被同一个包中的类所访问，包括它本身，但不能被别的包中的类不能访问。

（3）protected

protected可以用来修饰成员变量、成员方法、构造方法以及内部类。protected修饰的成员可以被同一个包内的其他类访问，不同包如果是它的子类的话也可以访问，其他包中不是它的子类的不能访问。

（4）public

public可以用来修饰成员变量、成员方法、构造方法以及类和接口。public修饰的成员可以被任何类访问，无论是否在同一个包内。

类和类成员都存在访问级别的定义。对于普通外部类来说，只有public和默认权限两种，普通外部类不能使用protected和private修饰符。

类的成员（包括内部类）的访问级别有四种：private、默认方式、protected、public。

4.类变量和实例变量有什么区别？

类的成员变量有2种：一种是类变量，一种是实例变量。定义成员变量时，若变量前有关键字static，则称为类变量，若没有static关键字，则称为实例变量。类变量存贮在类的公用区，属于类所有，所有该类的对象共享这一个类变量，而实例变量属于对象所有，每个对象拥有自已独立的实例变量，保存在对象的内存中。

5.类方法如何调用，是否必须创建对象才能调用类方法？

方法声明时，方法名前不加static的是实例方法，加static的是类方法。

类方法调用时不必创建对象就可以调用。

调用类方法有两种格式：

对象名.方法() 或者 类名.方法()

建议使用： 类名.方法();

6.实例方法如何调用？

对象名.方法();

7.final修饰类、变量、方法时有什么含义？

final修饰的类表示在该类不能有子类。

final修饰变量有3种情况：

final修饰的类变量只能被赋值一次，以后不能改变。要么定义时赋值，要么在类的初始化代码块中赋值，以后不能再赋值。

final修饰的实例变量只能被赋值一次，以后不能改变。要么定义时赋值，要么在构造方法中赋值，以后不能再赋值。

final修饰的局部变量只能被赋值一次，以后不能改变。要么定义时赋值，要么在后续代码中赋值一次，以后不能再赋值。

final修饰的成员方法表示在子类中不能被覆盖。

## 第5章 抽象类和接口

1.接口与抽象类有什么区别？什么情况下使用接口？什么情况下使用抽象类？

抽象类与接口紧密相关。然而接口又比抽象类更抽象，这主要体现在他们的差别上：

（1）类可以实现多个接口，但仅能从一个抽象（或任何其他类型）类继承，从抽象类派生的类仍可实现接口。接口可以支持多继承；

（2）抽象类中可以有普通方法，接口里的方法都是抽象方法，且都是public方法；

（3）抽象类中的成员变量可以被不同的修饰符来修饰，可以有实例变量，也可以有类变量，接口中的成员变量默认的都是public、static、final的变量；

（4）抽象类中有构造方法，接口中不能有构造方法，所以也不能创建对象。

抽象类是对象的抽象，里面的抽象方法也是对子类的一种规范，也可以有普通方法，这些普通方法子类可以继承使用，而接口只是一种行为规范。

那么什么情况该用抽象类，什么情况该用接口呢？主要是看现实的应用场景。如果把现实问题抽象出来后更符合类的特征，具有实例变量、非抽象的方法，那么只能是用抽象类；如果实际问题只能抽象出需要规范的功能方法，而且这些方法不知道如何实现，需要在子类再实现，不能抽象出实例变量，那么就应该使用接口。当然这时使用抽象类也可以，但是会丧失灵活性，因为其他类继承该抽象类后就不能再继承别的类了。

2.什么是多态？多态的好处是什么？多态有哪几种实现方式？

通俗的说，多态是指实现接口的多个类或一个父类的多个子类虽然有相同的方法，但是具有不同的表现方式。

Java中多态有两种实现形式。一种是通过方法的重载，一种是通过方法覆盖，即子类通过覆盖父类的方法或类实现接口的方法。

方法的重载是指一个类中定义了多个名字相同、参数不同的方法。当实际调用时，java编译器会根据参数的不同来自动决定调用哪个方法。

利用子类覆盖父类的方法或者利用实现接口的方法来实现的多态，是在程序运行期才能决定应该运行的哪个子类对象的方法，因而称为动态多态。

3.内部类有几种形式？

内部类有4种形式：实例成员内部类、静态内部类、局部内部类、匿名内部类。

4.外部类里面如何使用实例成员内部类？其他类里如何使用实例成员内部类？

在外部类里使用内部类跟使用普通类一样，直接调用即可。

但是在其他类里使用内部类有与使用普通类语法不一样，有单独的语法格式，需要分为两步，具体如下：

外部类 外部类对象名 = new 外部类构造方法(实参列表);

外部类.内部类 内部类对象名= 外部类对象名.new 内部类构造方法(实参列表);

5.在外部类里面如何使用static内部类？在其他类里如何使用static内部类？

在外部类中使用静态内部类跟使用普通类一样，直接调用即可。

在其他类中使用静态内部类，不需要创建外部类的对象，就可以创建内部静态类的对象。格式如下：

外部类.内部类 内部类对象名= new 外部类.内部类(实参列表);

6.内部匿名类如何定义类体和创建对象？

创建匿名内部类必须事先有一个接口或者父类，把内部匿名类定义为接口的一个实现类或者是某个父类的子类。实际上是把创建对象和定义匿名内部类的类体同时进行的。格式如下：

new 父类构造器（参数列表）{

//匿名内部类的类体部分

}

或者

new 接口（）{

//匿名内部类的类体部分

}

## 第6章 异常处理

一、问答题

1．什么是异常？异常分哪几种？

在编写程序过程中，会出现各种错误。有语法错误，有程序运行过程中出错，还有的程序运行过程中尽管没有报错，但出来的结果不对，这属于程序的逻辑错误。除了语法错误和逻辑错误外，例如，想打开的文件不存在、网络连接中断、操作数超出预定范围、正在装载的类文件丢失、访问的数据库打不开等等。Java把这些非正常的意外事件称为异常（Exception，又称为“例外”）。

系统定义异常：指java中对一些常见问题提供了对应的异常类。

用户自定义异常：程序员对特定问题，根据java规范编写的异常类。

Java中所有的异常类都是java.lang.Throwable的子类。Throwable类有两个直接子类：Error类及Exception类。

Error类描述的是内部系统错误，包括动态链接失败、虚拟机错误等，Java程序不做处理。这类异常主要是和硬件有关系，而不是由程序本身抛出。通常不由Java程序处理，用户也无法捕获。

Exception类是Java程序中需要大量处理的。

Java中定义了许多Exception的子类，这些子类分为两种：RuntimeException和非运行时异常。

运行时异常

运行时异常RuntimeException是一些可以通过适当的处理而避免的异常，在程序运行中可自动由JVM引发并处理，编程时不需捕获或声明。如：被0除，数组下标越界等，这类异常是编码时考虑不周产生的，完全可以通过判断来避免。

非运行时异常

非运行时异常，通常由环境因素引起的，与程序员无关，如输入、输出异常IOException，文件不存在、无效的URL等等。Java编译器要求Java程序必须捕获或声明所有的非运行时异常。

2．Java通过什么语句来捕获异常？

java可以通过try、catch、finally语句来捕获异常。

3．简述Java的异常处理机制

概括的说，java的异常处理机制就是三个步骤：

（1）异常的抛出。程序在执行过程中发生异常，java虚拟机就会发现并产生一个异常对象，这个过程就是异常的抛出。可以由系统来抛出异常，也可以由程序员在代码中使用throw强制抛出某种类型的异常。

（2）异常的捕获。出现异常后，java可以通过try、catch、finally语句来捕获异常，并做相应的处理，这个过程称为异常的捕获。

（3）如果没有捕获异常的代码，程序将终止运行。

4．简述try、catch、finally的使用方法

try-catch-finally基本格式：

try{

//可能产生异常的代码放在此处

}

catch (异常类名1 异常对象名）{

//在此处理异常类型1对应的异常

}

catch (异常类名2 异常对象名）{

//在此处理异常类型2对应的异常

}

finally{

//最终的程序出口，往往会写一些清理资源的语句放在此处

}

try-catch-finally语句把可能产生异常的语句放入try{ }语句块中。当try代码块中的语句发生了异常，系统将这个异常发生的代码行号、类别等信息封装到一个对象中，并将这个对象传递给catch代码块。程序就会跳转到catch代码块中执行。

在try{ }语句后紧跟一个或多个catch块，用于对try代码块中所生成的异常对象进行处理，每一个catch块处理一种可能抛出的特定类型的异常。在运行时刻，如果try{ }语句块中产生的异常与某个catch块处理的异常类型相匹配（匹配是指异常参数类型与实际产生的异常类型一致或是其父类），则将停止执行try中的剩余语句，并跳转到catch中执行该catch语句块。

finally语句是捕获异常的最后一步，是通过finally语句为异常处理提供一个统一的出口。finally子句是可选的。无论try所指定的程序块中是否抛出异常、抛出哪种异常，finally关键字中包含的代码都要被执行。

二、编程题

练习自定义异常的用法。定义一个年龄异常类AgeException，定义一个Person类，具有name（姓名）、age（年龄）属性，以及setAge（int age）方法，当给age属性的值不在0-200范围内时抛出年龄异常。

public class AgeException extends Exception{

public AgeException(){

//调用父类的构造方法，将"年龄超过范围"赋值给父类的message变量

super("年龄超过范围");//异常错误信息是固定的

}

public AgeException(String message){

//调用父类的构造方法，将message传递给父类的message变量

super(message);//异常错误信息由调用者指定

}

}

public class People {

private String name;//姓名

private int age;//年龄

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public int getAge() { return age; }

public void setAge(int age) throws AgeException{

if(age<0||age>200)

throw new AgeException();

this.age = age;

}

public static void main(String[] args) {

People p=new People();

try {

p.setAge(400);

} catch (AgeException e) {

System.out.println("给age赋值错误："+e.getMessage());

}

}

}

## 第10章 图形界面

一、问答题

1．java有哪些常用的布局？每种布局的特点是什么？

主要有FlowLayout、BorderLayout、GridLayout、CardLayout 、BoxLayout等布局管理器类以及空布局。

FlowLayout布局也称为流式布局，他是JPanel的默认布局方式。他是按照组件添加的先后顺序，默认从左到右依次将组件添加到容器中，根据容器宽度，放满一行后，自动从下一行开始由左到右排列组件。

BorderLayout是一个边框布局，他是Window类型的容器的默认布局，例如JFrame、JDialog的默认布局都是BorderLayout。他可以对容器组件进行安排，并调整其大小，使其符合下列五个区域：北区、南区、东区、西区、中区。每个区域只能包含一个组件，并通过相应的常量进行标识：NORTH、SOUTH、EAST、WEST、CENTER。

GridLayout是一个网格布局，按照多行多列的方式排列组件，每个行列的交叉区域称为一个单元格，每个组件占据一个单元格，并根据单元格的大小调整其大小。

CardLayout 称为卡片布局。它将容器中的多个组件像扑克牌一样层叠在一起，每个组件被当做一张卡片。第一个加到容器的组件在最上面，第二个加入的组件在第一张的下面，依次顺序排列，最后加入的组件在最下面。同一时刻只能显示一个组件。

空布局有时也叫自由布局。空布局其实就是不指定布局，通过指定各个组件的坐标来指定组件的绝对位置。

2．FoucusListener中声明了哪几种方法？什么情况下执行哪个方法？

FocusListener接口中只声明了2个方法：

void focusGained(FocusEvent e)：事件源组件获得键盘焦点时调用。

void focusLost(FocusEvent e)：事件源组件失去键盘焦点时调用。

3．MouseListener中声明了哪几种方法？什么情况下执行哪个方法？

在事件源上发送以下5种动作时需要使用MouseListener接口来处理：按下鼠标键、释放鼠标键、单击鼠标、鼠标进入、鼠标离开。接口中声明的5个方法是：

public void mouseClicked(MouseEvent e)：鼠标在组件上单击时

public void mousePressed(MouseEvent e)：鼠标组件上按下时

public void mouseReleased(MouseEvent e)：鼠标组件上释放时

public void mouseEntered(MouseEvent e)：鼠标进入组件时

public void mouseExited(MouseEvent e)：鼠标离开组件时

4．实现监听器有哪3种方法？

监听器的编写有三种方法：

（1）编写一个单独的类实现某个监听器接口或继承某个监听器适配器类。

（2）让窗口类直接实现相应的接口。

（3）用匿名类的方式直接继承某个监听器适配器类或实现某个监听器接口（最常用的方法）。

二、选择题

1．获取JTextField组件中输入的文本值的方法是：（A）

A. getText() B.setText() C. getLabel() D setLabel()

2．获取一个复选框是否选中的方法是：（A）

A. isSelected() B. isChecked() C. setChecked() D. setSelected()

3．给一个按钮注册动作监听器的方法是：（B）

A. addActionEvent() B. addActionListener()

C. addMouseListener() D. addItemListener()

三、编程题

1．设计一个用户注册的界面，包括用户名、密码输入框，性别单选按钮，表示爱好的篮球、足球、乒乓球复选框，表示学历的下拉列表框，注册按钮和取消按钮。

public class Register extends JFrame {

//定义要在窗口中显示的属性，即组件

JLabel lblUserName=new JLabel(" 用户名：");

JTextField txtUserName=new JTextField(10);

JLabel lblPassword=new JLabel(" 密码：");

JPasswordField txtPassword=new JPasswordField(10);

JLabel lblSex =new JLabel(" 性别：");

ButtonGroup group=new ButtonGroup();

JRadioButton rbtMale=new JRadioButton("男",true);

JRadioButton rbtFeMale=new JRadioButton("女",false);

JLabel lblHobby =new JLabel(" 爱好：");

JCheckBox cbxBasketball=new JCheckBox("篮球",true);

JCheckBox cbxFootball=new JCheckBox("足球");

JCheckBox cbxTableTennis=new JCheckBox("乒乓球");

JLabel lblXueLi =new JLabel(" 学历：");

JComboBox jbxXueLi=new JComboBox();

JButton btnOk=new JButton("确定");

JButton btnCancel=new JButton("取消");

JPanel p=new JPanel();JPanel pJbxx=new JPanel();

public Register(){

//将组件放入窗口中，基本信息放在上边的面板pJbxx中，简历放在下边的面板pJianLi中

pJbxx.setLayout(new GridLayout(0,4));//每行放4个组件

//学号和密码占一行

pJbxx.add(lblUserName);pJbxx.add(txtUserName);

pJbxx.add(lblPassword);pJbxx.add(txtPassword);

txtPassword.setEchoChar('A');txtPassword.setText("abcd");

//性别占一行，不够4个，用一个用空的JLabel填充

pJbxx.add(lblSex);pJbxx.add(rbtMale);pJbxx.add(rbtFeMale);

pJbxx.add(new JLabel());

//把男女单选按钮放到一个组里，实现互斥

group.add(rbtMale);group.add(rbtFeMale);

//学院占一行，该行不够4个，用2个空的JLabel填充

pJbxx.add(lblXueLi);pJbxx.add(jbxXueLi);

pJbxx.add(new JLabel());pJbxx.add(new JLabel());//

jbxXueLi.addItem("博士");//添加下拉选择项

jbxXueLi.addItem("硕士");

jbxXueLi.addItem("本科");

jbxXueLi.addItem("专科");

//爱好占一行

pJbxx.add(lblHobby);pJbxx.add(cbxBasketball);

pJbxx.add(cbxFootball);pJbxx.add(cbxTableTennis);

//确定和重置按钮占一行，该行不够4个，用2个空的JLabel填充

pJbxx.add(new JLabel());pJbxx.add(btnOk);pJbxx.add(btnCancel);

pJbxx.add(new JLabel());

p.add(pJbxx);

add(p,BorderLayout.CENTER);

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

}

public static void main(String[] args) {

Register frame=new Register();

frame.setSize(500,300);

frame.setVisible(true);

}

}

2．设计一个求三角形面积的图形界面程序，要求通过3个输入框输入3个边长，当单击“计算”按钮时，能够计算以输入的3个边长所形成的三角形的面积。要求能够判断输入的3个边长是否是数字，三个边长是否能够构成合法的三角形。

public class TriangleFrame extends JFrame{

JLabel lblEdge1=new JLabel("边长1");

JLabel lblEdge2=new JLabel("边长1");

JLabel lblEdge3=new JLabel("边长1");

JLabel lblArea=new JLabel("面积");

JTextField txtEdge1=new JTextField(10);

JTextField txtEdge2=new JTextField(10);

JTextField txtEdge3=new JTextField(10);

JTextField txtArea=new JTextField(10);

JButton btnCalculate=new JButton("计算");

JButton btnReset=new JButton("清空");

JPanel p=new JPanel();

public TriangleFrame(){

super("计算机三角形面积");

p.setLayout(new GridLayout(5,2));

p.add(lblEdge1);p.add(txtEdge1);

p.add(lblEdge2);p.add(txtEdge2);

p.add(lblEdge3);p.add(txtEdge3);

p.add(btnCalculate);p.add(btnReset);

p.add(lblArea);p.add(txtArea);

this.add(p);

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

btnCalculate.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

double a=Double.valueOf(txtEdge1.getText());

double b=Double.valueOf(txtEdge2.getText());

double c=Double.valueOf(txtEdge3.getText());

if(!(a+b>c&&a+c>b&&b+c>a)){

JOptionPane.showMessageDialog(null, "您输入的3个边长不能构成一个三角形","提示信息",JOptionPane.ERROR\_MESSAGE);

return;

}

Triangle triangle=new Triangle(a,b,c);

txtArea.setText(String.valueOf(triangle.getArea()));

}

});

btnReset.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

txtEdge1.setText("");

txtEdge2.setText("");

txtEdge3.setText("");

}

});

}

public static void main(String[] args) {

TriangleFrame frame=new TriangleFrame();

frame.setSize(300,300);

frame.setVisible(true);

}

}

class Triangle {

private double a,b,c;

public double getA() {

return a;

}

public void setA(double a) {

this.a = a;

}

public double getB() {

return b;

}

public void setB(double b) {

this.b = b;

}

public double getC() {

return c;

}

public void setC(double c) {

this.c = c;

}

public Triangle(double a, double b, double c) {

super();

this.a = a;

this.b = b;

this.c = c;

}

public double getArea(){

double p=(a+b+c)/2;

return Math.sqrt(p\*(p-a)\*(p-b)\*(p-c));

}

}

**复习知识点总结：**

1. **JAVA程序的运行，Java源文件的组成；**
2. **标识符的命名规则；**
3. **面向对象程序的三大主要特征：封装、继承和多态；**
4. **方法的参数传递**
   1. 基本数据类型：当将基本数据类型作为参数，传递给方法，方法运行结束后，原值不会发生改变；
   2. 引用数据类型：当将引用数据类型作为参数，传递给方法，方法运行结束后，原值会发生改变；
5. **方法的重载**
6. **构造器（构造方法）的声明和使用**：
   1. 构造器的名称必须与类名一致；
   2. 若一个类中没有显示提供任何构造器时，系统会提供一个默认无参构造器；
   3. 若一个类中提供了任何构造器，系统默认无参构造器将不再提供；
   4. 构造器只能调用一次，并且是在创建对象时；
   5. 构造器没有返回值类型，也不需要 return；
   6. 构造器可以构成重载；
7. **继承的使用**
   1. java中支持的是单继承，不支持多重继承， 但支持多层继承；
   2. 子类可以继承父类的所有属性和方法，只是私有的属性，不能直接访问，而是通过公共的setXXX或getXXX方法访问;
8. **方法的重写（覆盖，覆写）**
9. **final关键字的使用**
   1. 修饰类时不能被继承；
   2. 用于修饰方法，此方法不能被重写，但可以被继承；
   3. 用于修饰变量（属性或局部变量），变成了常量，赋值后，值不能更改！！！
   4. 修饰属性，必须在声明时或构造器或初始化块赋值，赋值后不能更改；
10. **对象的初始化顺序**
11. **this和super关键字的使用**
    1. **this：**使用在本类中，代表当前对象。可用于调用 属性、方法、构造器
    2. **super：**
       1. 父子类的成员名相同时，在子类中默认使用的是子类的成员（就近原则），假如说想访问父类的成员，可以采用super关键字调用。
       2. super关键字代表的是父类的引用，可以访问父类的属性、方法、构造器。
12. **子类调用父类构造器的方法：**子类不能继承父类的构造器，但是必须调用父类的构造器。
    1. 子类的所有构造器，都默认调用 父类的无参构造器；
    2. 如果父类没有无参构造器。子类必须显式的调用父类的有参构造器，否则报编译错误！！
13. **Java中流程控制语句的种类**
14. **数组的声明，初始化和使用**
15. **Static关键字（修饰类，方法和属性时的不同特点）**
16. **成员变量和局部变量**
17. **实例方法和类方法**
18. **静态代码块**
19. **抽象类和接口的定义使用**

关键字 有无构造 实例化 继承 抽象方法 普通成员

**普通类** class 有 可以 单继承 无 有

**抽象类** abstract class 有 不可以 单继承 有 有

**接口** interface 无 不可以 多继承 必须有 必须无

1. **匿名内部类的定义使用**
2. **多态的体现**
   1. 重载和重写 （方法的多态）
   2. 对象的多态性（对象的转型）（对象的多态）
      1. 向上转型
      2. 向下转型
3. **异常处理方式**: try-catch-finally ,throws，throw的使用
4. 熟悉常用的顶层容器，中间层容器以及组件命名和简单使用；
5. 几种常用的布局管理器，顶级容器以及面板组件的默认布局管理器
6. 常用的事件监听器，事件监听器的注册方式；

**注：对照知识点，通过教材和课件认真复习！！！**

**考点总结：**

1. **JAVA程序的运行，Java源文件的组成；**
2. **标识符，关键字，注释；**
3. **面向对象程序的三大主要特征：封装、继承和多态；**
4. **方法的参数传递**
5. **方法的重载**
6. **构造器（构造方法）的声明和使用**：
7. **继承的使用**
8. **方法的重写（覆盖，覆写）**
9. **final关键字的使用**
10. **this和super关键字的使用**
11. **Java中流程控制语句的种类**
12. **数组的声明，初始化和使用**
13. **类变量和实例变量**
14. **实例方法和类方法**
15. **抽象类和接口的定义使用**
16. **匿名内部类的定义使用**
17. **多态的体现：向上转型**
18. **异常处理方式**: try-catch-finally ,throws，throw分别什么情况下使用。
19. 熟悉常用的顶层容器，中间层容器以及组件命名和简单使用；
20. **布局管理器**：顶级容器以及面板组件的默认布局管理器
21. **事件监听器：**事件监听器的注册方式；

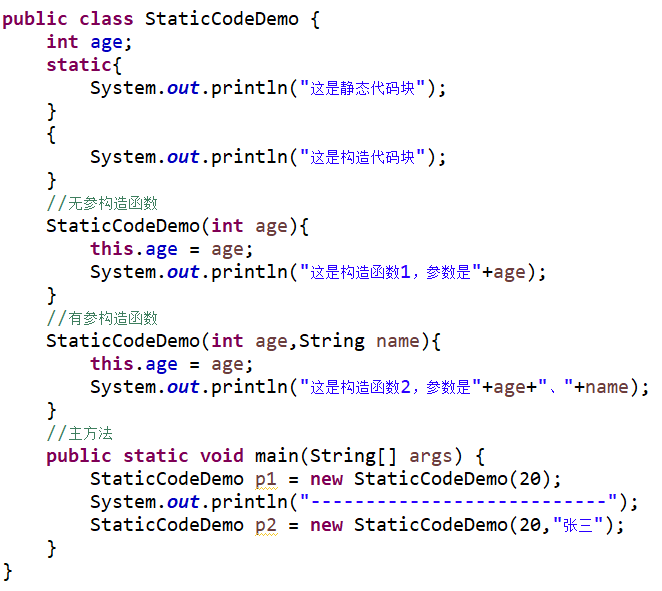
**22：对象初始化顺序**

**静态代码块和构造代码块（构造代码块执行在静态代码块之后，在构造方法之前）**

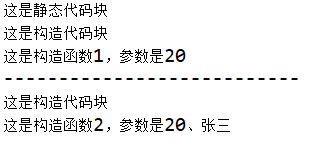
**构造代码块：** 对象实例化就执行，每实例化一次执行一次

**静态代码块：** 对象实例化就执行，无论new多少个对象，仅执行一次；

**初始化顺序：**静态成员变量及static代码块——普通成员变量及普通代码块——构造方法——。。。



**运行结果：**



**注：仔细看3~5章，书上和课件的知识点以及程序代码！**