Javap 反编译用的，implements A,B,C,D

Array 可以 int []arr[] = new int[10][10] //逆天写法。。

变量 byte（1 个字节 8 位） short（2 个字节 16 位） int（4 个字节 32 位） long（8 个字节 64 位） float（4 个字节 32 位） double

精度顺序byte, short, int, long, float, double ，从小到大自动转换，从大到小要显式转换

char 转 short 要显式转换

3.402 是double 类型的 float 类型的后面要加f/F Scanner reader=new Scanner(System.in);

Reader 可以调用方法 nextByte(),nextShort(),nextInt()…..读取用户输入的各种数据用户需要输入数据回车确认

While(reader.hasNextDouble())等待用户从键盘输入数据 X=reader.nextDouble();读取数据

System.arraycopy(copyFrom, 2, copyTo, 0, 7) 将数组 copyFrom 中的元素从 2 开始对应拷贝到 copyTo 数组的 0 到 7

<<算数左移 <<<逻辑左移

Switch（）中表达式的值必须是整型或字符型

Case 常量值 1：常量值也必须是整型或字符型 ； 'Default 可有可无

第四章

封装：将数据和对数据的操作封装在一起（类）

继承：子类可以继承父类的属性和功能，同时可以增加子类独有的属性和功能

多态：1 多个操作有相同的名字，但操作接收的消息类型不同；2 同一操作被不同类型的对象调用时产生不同行为

类的名字可以有字母，_，$，数字 第一个字符不能是数字

Java 不提供默认的构造方法如果类中只写了一个有参构造，那么rectangleOne = new Rect();非法

成员变量会获得自己的内存空间——该对象的实体或变量

变量

Static 静态变量/类变量，其他称作实例变量 静态变量可以通过类名直接访问

final 只能赋值一次，但不一定要在声明的时候赋值

方法重载——一个类中可以有多个方法具有相同的名字，方法的参数必须不同，返回值可以不同

this

基本类型数据的类包装Java.lang 包中

比如 Character.isLowerCase(a[i]) 将字符数组中的字母都变为小写

第五章

Protected 不仅是子类可以访问，同个包内的也可以访问（surprise! 没错 protected 比default package private 还宽松）其他与 C++一致

父类的构造方法不会被子类继承。 没有ctor 的会自动获得一个空ctor()，有定义 ctor 的不会获得。

成员变量的隐藏

当子类中定义和父类中同名的成员变量时，子类隐藏了继承的成员变量

方法的隐藏——重写（名字、返回类型、参数个数及类型必须与父类一样）override重载（参数个数或类型不一样）overwrite

super 关键字

子类中使用 super 调用父类的构造方法、调用被子类隐藏的成员变量和方法

因为子类不继承父类的构造方法，因此如果子类想使用父类的构造方法，必须在子类的构造方法中使用 super，且 super 必须是子类构造方法中的第一条语句

Final 类不能被继承，final class A{ }. 因为 final 不可更改，因此子类中不能重写final

多态：父类的某个方法被其子类重写时，可以产生自己的功能行为

抽象类 abstract：

如果写了构造方法，最好用protected 修饰，因为是给子类用的不能用new 创建对象，必须由其子类创建对象

如果类体中有abstract 方法，只允许声明，不能实现非abstract 子类必须实现abstract 方法

Java 一个类只能继承一个父类，但是可以实现多个接口 interface

接口的使用 class A implements B, C

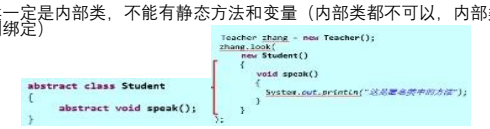
接口中，常量为public static final（可省略）；方法为 public abstract（可省略）接口也可以被继承 extends

接口回调：把实现某一接口的类创建的对象的引用赋给该接口声明的接口变量内部类：在一个类中声明另一个类；外嵌类：包含内部类的类

内部类中可以用外嵌类的成员变量和方法匿名类：把类体与对象的创建组合在一起

因为没有名字，所以不能声明对象——相当于没有声明直接向方法的参数传值，创建了一个对象

匿名类一定是内部类，不能有静态方法和变量（内部类都不可以，内部类实例和创建它时的外部类实例绑定）



（不止 abstract class，interface 也可以）

自定义异常类——需要extends 继承Exception 类，或继承class Throwable（它不是interface，Exception 本身extends Throwable）

范型写法：

```
class Chorus {
        void makeChorus(E person, F instrument) {
                person.toString();
                instrument.toString(); } }
```

第六章

s1.length() s1.equals(s2)比较两个字符串实体是否相同 s1.startsWith(s2)，endsWith(s2)

s1.compareTo(s2) s1 比 s2, 大于返回正值，小于返回负值 s1.indexOf(s2)

s1.substring(int start) 获得字符串s1 的[start, len)子串 s.replaceAll(s1,s2) 替换 s.trim() 去掉前后空格

Integer.parseInt(s) 将s 转为 int 类型 （parseByte Short Long Float Double）

String.valueOf(i) 将数字i 转为字符串

Java 可以直接 int+字符串，会自动把 int 类型转换成字符串（其他数字类型同，有 toString 的对象同） String.getChars(start, end, c[], offset) 将字符串中从位置int start 到int end-1 位置上的字符复制到数组c 的offset 处 char a[] = s.toCharArray();并将字符串对象的全部字符复制到该数组中（byte[] 是 .getBytes()

StringBuffer 线程安全，StringBuilder 线程不安全 默认初始容量为 16

StringBuffer(String s) 初始容量为参数字符串 s 的长度再加 16 个字符 sb.capacity()获取当前的实际容量

append 方法将其它 Java 类型数据转化为字符串后，再追加 到 StringBuffer 对象中

char charAt(int index)：得到参数 index 指定的位置上的单个字符（第一个位置为 0）

void setCharAt(int index, char ch) 将当前 StringBuffer 对象实体中的 字符串位置index 处的字符用参数 ch 指定的字符替换

StringBuffer insert(int index, String str)：将一个字符串插入另一个 字符串中，并返回当前对象的引用

StringBuffer insert(int index, String str)：将一个字符串插入另一个字符串中，并返回当前对象的引用

StringBuffer delete(int startIndex, int endIndex)：从当前 StringBuffer 对象实体中的字符串中删除一个子字符串（从 startIndex

到endIndex-1），并返回当前对象的引用

StringBuffer replace(int startIndex, int endIndex, String str)：将当前 StringBuffer 对象实体中的字符串的一个子字符串用参数 str

指定的字符串替换

StringTokenizer 需要将字符串分解成可被独立使用的单词时

StringTokenizer(String s, String delim)：为字符串s 构造一个分析器，参数delim 中的字符被作为分隔符 例如：StringTokenizer fenxi = new StringTokenizer(s,".")

字符串分析器可以使用nextToken()方法逐个获取字符串分析器中的语言符号（每获取一个 计数变量自动-1）字符串分析器调用 countTokens()方法可以得到计数变量的值

用循环逐个获取语言符号 while（s.hasMoreTokens())

Scanner 类 可创建出用于解析字符串的对象 Scanner scanner = new Scanner(cost);

(scanner.useDelimiter("[^0123456789.]+");通过正则表达式作为分隔标记) while(scanner.hasNext())

{

String t = scanner.next();

}

模式匹配

第一步，使用 Pattern 类创建一个对象，模式对象

Pattern p = Pattern.compile("A\\d");\\d 代表 0 到 9 中的任何一个第二步，建立匹配对象

p.matcher(CharSequence input); matcher 对象的常用方法

public boolean find(int start)：判断input 从参数 start 指定位置开始是否有和pattern 匹配的子序列

public String replaceAll(String replacement)：把 input 中与 pattern 匹配的子字符串全部替换为参数 replacement 指定的字符串

public String replaceFirst(String replacement)：把 input 中第一个 与 pattern 匹配的子字符串替换为参数 replacement 指定的字符串

Pattern p;

Matcher m;

String input = "0A1A2A3A4A5A6A7A8A9"; //被匹配的内容

p = Pattern.compile("\\dA\\d"); //要匹配的部分

m = p.matcher(input); while(m.find())

{ String str = m.group(); //返回匹配的字符串}

正则表达式

以使用一对方括号括起若干个字符，代表方括号中的任何一个字符

[abc]：代表 a, b, c 中的任何一个

[^abc]：代表除了 a, b, c 以外的任何字符

[a-d]：代表 a 至d 中的任何一个

X? 表示 X 出现 0 次或 1 次；X* X 出现 0 次或多次；X+ X 出现 1 次或多次X{n,} X 至少出现n 次；

\\D 代表任何一个非数字字符；\\d 代表 0~9 的任何一个数字；\\S 代表非空格类字符；\\w 代表可用于标识符的字符（不包括美元符号）；\\W 代表不能用于标识符的字符；\\p{Alpha}表示字母字符

String str = "Please logon :http://www.cctv.cn Watch TV"; String regex = "(http://|www)[.]?\\w+[.]{1}\\w+[.]{1}\\p{Alpha}+";

第七章

Date 类——获取本地当前时间

创建对象：Date(long time) time 取值为毫秒 60*60*1000 public long currentTimeMillis()获取系统当前时间，epoch 毫秒数

可以使用 DateFormat 的子类 SimpleDateFormat 来实现日期的格式化：

SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy-MM-dd ");

将时间对象转为字符串

String currenTime = sdf.format(sdf1);

Calendar 类

Calendar calendar = Calendar.getInstance(); 初始化一个日历对象 calendar.setTime(new Date()); 将日历翻到一个时间

String day_of_week = String.valueOf(calendar.get(Calendar.DAY_OF_WEEK)-1);//获取星期 注意要-1 获取月份 calendar.get(Calendar.MONTH);

Math 类

public static double random()：产生一个 0 到 1 之间的随机数，范围 是[0,1)

BigInteger 类

add(b)、subtract(b)、multiply(b)、divide(b)、remainder(b)、compareTo(b)（大等小于依次返回 1,0,-1），bi.pow(exp) toString()、toString(int p) p 为进制

数字格式化

用 NumberFormat 类

NumberFormat f = NumberFormat.getInstance();//实例化对象

可调用 public final String format(double number)方法来格式化数字 number f.setMaximumFractionDigits(5); //设置小数的最大位数 f.setMinimumIntegerDigits(3);//设置整数的最小位数

LinkedList<E>泛型类 创建链表结构的数据对象

LinkedList<String> mylist = new LinkedList<String>();

Add()向链表末尾增加节点 addFirst()头加 removeFirst() removeLast() get() getFirst() getLast() indexOf() lastIndexOf()

关键字

abstract, continue, for, new, switch, assert***, default, goto*, package, synchronized, boolean, do, if, private, this, break, double, implements, protected, throw, byte, else, import, public, throws, case, enum****, instanceof, return, transient, catch, extends, int, short, try, char, final, interface, static, void, class, finally, long, strictfp**, volatile, const*, float, native, super, while • * not used • ** added in 1.2 • *** added in 1.4 • **** added in 5.0

按取值范围从"低"到"高"排列– byte, short, int, long, float, double •

当把级别低的变量的值赋给级别高的变量时，系统自动完成数据类型的转换 如int型转换成long型

set(index, elem) 将当前链表 index 位置节点中的对象替换为参数 element 指定的对象，并返回被替换的对象

size() contains(obj) clone()

iterator 用法：Iterator<Integer> container.iterator()，然后 while(iter.hasNext()) { iter.next() }

HashSet<E> 创建的对象称作集合，add 进集合中的元素称作元素

集合对象的容量会随着添加元素而增加：若集合添加的元素超过总容量的 75%，集合容量加一倍 Add(obj) clear() contains(obj) isEmpty() remove(obj) toArray()

containsAll(HashSet set) 是否包含另一个集合中的所有元素

foreach 遍历：

HashMap<String, Integer> hm = new HashMap<>();

for(Map.Entry<String, Integer> e : hm.entrySet()) { // 还有keySet()、values()

}

public Object clone()：得到当前集合一个克隆对象。注意 Clone()返回的是 Object 要强制类型转换

boolean addAll(HashSet set) 并//retainAll(HashSet set) 交// removeAll(HashSet set) 差

HashSet 实现了 Set 接口，Set 接口继承自Collection 接口

HashMap<k,v>泛型类 散列映射

remove() clear() clone() containsKey(obj) containsValue(obj) get(key) isEmpty() size() put(k,v)不是 add()

TreeSet<E>泛型类 树集

比hashmap 多一个 first() last()

Student 类中可以重写方法compareTo 自定义排序方法(implements Comparable)

方法一：实现 Comparator 接口，创建比较器（是接口不是类！ implements Comparator）

compare(Object o1,Object o2) 是接口中的方法 new XXXComparator()作为 Arrays.sort()的第二参数

方法二：实现 Comparable 接口，实现 compareTo(Object o2) TreeMap<K.V>(Comparator<K> comp)按 comp 接口规定的大小来排序

iterator 遍历treemap：

Collection<Student> collection = treemap.values(); //和Hash Iterator iter = collection.iterator();

while(iter.hasNext()) { Student te=iter.next(); }

Stack 泛型类

Push() pop() empty() peek() Boolean empty()

第 8 章

线程的 4 种状态（完整的生命周期）：

新建 此时，已经有了相应的内存空间和其他资源

运行 线程创建后仅占有了内存资源，需要调用 start()通知 JVM，告诉 JVM 有个新的线程排队等候切换；JVM 将CPU 使用权切换给线程时，如果线程是 Thread 的子类，run()方法立刻执行。必须在子类中重写 run 方法； 线程没有结束 run()方法之前，不要能再调用 start()方法

中断 1 JVM 将CPU 资源从当前线程切换给其他线程 2 线程执行了 sleep(int millsecond)——重新进到线程队列中排队等待 CPU 资源 3 执行了 wait()方法，当前线程进入中 断（等待）状态，等待状态的线程不会主动进到线程队列中排队等待 CPU资源，必须由其他线程调用notify()方法通知它

死亡——释放了线程对象的内存 1 执行完 run()方法中的全部语句 2 stop()被提前强制终止，即强制 run()方法结束让线程主动调用 sleep 方法让出 CPU 的使用权

使用 wait()方法可以中断方法的执行，使本线程等待，暂时让出 CPU的使用权，并允许其它线程使用这个同步方法。其它线程如果在使用这个同步方法时不需 要等待，那么它使用完这个同步方法的同时，应当用 notifyAll()方法通知所有由 于使用这个同步方法而处于等待的线程结束等待。线程联合：B.join()如果线程 A 在占有 CPU 资源期间一旦联合 B 线程，那么 A 将立刻中断执行，一直等到它联合的线程 B 执行完毕，A 线程再重新排队等待 守护线程：当程序中的所有用户线程都已结束运行时，即使守护线程的 run() 方法中还有需要执行的语句，守护线程也立刻结束运行 thread.setDaemon(true);

JVM 中的**线程调度器**负责**管理线程** 把线程的优先级 10 个级别优先级都在常数 1 到常数 10 的范围内；如果没有明确设置优先级默认为 5；优先级可以通过 setPriority(int grade)方法调整，getPriority 方法返回

JVM 中的线程调度使高优先级的线程能始终运行（若A 和B 的级别高于 C 和D，等到 A 和B 都执行完毕进入死亡状态，才会在 C 和D 之间轮流切换）

t.join()："进程联合"创建线程的途径

用Thread 子类创建线程 class Left extends Thread

用 Thread 类直接创建线程对象 构造方法中的参数是一个 Runnable 类型的接口，因此，在创建线程对象时必须向构造方法的参数传递一个实现 Runnable 接口的类的对象（先创对象，再将对象传给 thread 对象）

Implements Runnable，实现 public void run();

Extends Thread，也是实现 public void run();

currentThread()方法是Thread 类中的静态方法，可以用类名调用，该 方法返回当前正在使用 CPU 资源的线程

第十章 URL 网络编程 URL 对象通常包含最基本的三部分信息：协议、地址、资源。 常用的 http、ftp、file 协议都是 Java虚拟机支持的协议 地址必须是能连接的有效的 IP 地址或域名（host name） 资源可以是主机上的任何一个文件 url = new URL(text.getText().trim()); InputStream in = url.openStream(); while((n=in.read(b)) != -1){ String s = new String(b,0,n); area.append(s);} catch(MalformedURLException e1) catch(IOException e1) 显示 HTML 文件 public JEditorPane(URL initialPage) throws IOException 处理超链 HyperlinkListener 方法 void hyperlinkUpdate(HyperlinkEvent e) InetAddress 类的对象含有一个 Internet 主机地址的域名和 IP 地址，例如： www.sina.com.cn/202.108.35.210 套接字 Socket ： 端口号与 IP 地址的组合得出一个网络套接字。

服务端

ServerSocket server = new ServerSocket(4333) Socket s = server.accept();

in = new DataInputStream(socketAtServer.getInputStream()); // in.readInt、readLong、readFloat、readDouble、readUTF out= new DataOutputStream(socketAtServer.getOutputStream());//writeInt、writeLong、etc

out.writeUTF("") in.readUTF();

客户端 Socket s = new Socket("localhost", 4333)

数据报：

ds = new DatagramSocket(23333);

DatagramPacket p = new DatagramPacket(buf, BUFFER_SIZE) ds.receive(p); // throws IOException

发送：

ds.send(new DatagramPacket(sendbuf, sendbuf_size, InetAddress.getByName("127.0.0.1"), 23333));

第八章流

java.io 中有 4 个重要的 abstract class InputStream（字节输入流）OutputStream（字节输出流）Reader（字符输入流） Writer

（字符输出流） File read2 = new File("C:\\Users\\slark ","userTotTime.txt"); //文件路径 try { read2.createNewFile(); //新建文本文件 }catch(IOException ex) {} 用 scanner 解析 Scanner scanner = new Scanner(file); 那么 scanner 将空格作为 分隔标记、调用 next()方法依次返回 file 中的单词。会抛出 IOException。

用法： while(scanner.hasNext()){scanner.nextInt()} nextInt 有 InputMismatchException 异常不想是空格也可以正则表达式 scanner.useDelimiter(正则表达式); 如="[^0123456789.]+"解析出来就只剩下数字。 模板： try{BufferedReader input = new BufferedReader(new FileReader("input.txt")); BufferedWriter output = new BufferedWriter(new FileWriter("output.txt")); Str ing s=null; int i=0; while((s = input.readLine())!=null){ i++; output.write(i + ": " + s); output.newLine(); } output.flush(); output.close(); input.close(); fw.close(); fr.close(); }catch(IOException e){ } 文件字节流 FileOutputStream(String name||File file) write(byte b[],

（of, int len）) 数据流：DataIn（Out）putStream FileOutputStream fos = new FileOutputStream("jerry.dat"); DataOutputStream output = new DataOutputStream(fos); output.writeInt(100); output.writeChars("I am ok"); 对象 流： ObjectInputStream writeObject（obj）不过该类要实现 Serializable 接 口才能被序列化，不用实现额外方法 ByteArrayOutputStream out = new ByteArrayOutputStream() ObjectOutputStream objectOut = new ObjectOutputStream(out); objectOut.writeObject(shop1); ByteArrayInputStream in = new ByteArrayInputStream(out.toByteArray()); ObjectInputStream objectIn = new ObjectInputStream(in); Shop shop2 = (Shop)objectIn.readObject();

第九章 图形用户界面设计 java.awt 包中的类创建的组件习惯上称为重组件。创建一个按钮组件时，都有 一个相应的本地组件在为它工作，即显示主件和处理主件事件，该本地主件称 为它的同位体。 javax.swing 包为我们提供了更加丰富的、功能强大的组件，称为 Swing 组件， 其中大部分组件是轻组件，没有同位体，而是把与显示组件有关的许多工作和 处理组件事件的工作交给相应的 UI 代表来完成。 这些 UI 代表是用 Java 语言编写的类，这些类被增加到 Java 的运行环境中，因此组件的外观不依赖平台，不仅在不同平台上的外观是相同的，而且与重组件相比有更高的性能。 JComponent 类的子类都是轻组件 JFrame, JApplet, JDialog 都是重组件，即有同位体的组件。这样，窗口（JFrame）、小应用程序（Java Applet）、对话框（JDialog）可以和操作系统交互信息。轻 组件必须在这些容器中绘制自己，习惯上称这些容器为 Swing 的底层容器。 对于 JFrame 窗口， 默认布局是 BorderLayout 布局 FlowLayout 、BorderLayout 、CardLayout 、GridLayout 布局 null 布局 p.setLayout(null); setBounds(int a, int b, int width, int height) JButton next = new JButton("确定(5s)"); //确认按钮 Font style = new Font("宋体", Font.BOLD, 24); //字体格式：宋体 24 号 JLabel A, B, C, D, question, tips, rate, fin; //框架中的文字， 用 JLabel JTextField inputans; //输入框 setTitle(s); setLayout(null); translate1.setFont(style); setBounds(500, 250, 500, 300); //给整个窗口设置 位 置 大 小 setVisible(true); validate(); setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); public void actionPerformed(ActionEvent e) {e.getSource() == translate1 }

初始化线程、事件调度线程、工作线程（自己创建）， MouseAdaptor：类，MouseListener：接口。前者为后者的一个实现，提供了空实现。

Scanner: 有 nextByte(), nextShort(), nextInt(), nextLong(), nextFloat(), nextDouble(), nextLine() 、hasNext() 、

hasNextDouble()

BufferedReader 有 readline()

new DataInputStream(InputStream is)有 readByte(), readShort(), readInt()……

new DataOutputStream(OutputStream os)有 writeByte(), writeShort(), writeInt()……读入用Scanner in = new Scanner(System.in)，然后 while(hasnextxxx) {nextxxx()}

Trick：可以 catch (InputMismatchException)里调用 scanner.next()来跳过不符合的部分

Synchronized

线程池：

ExecutorService executor = Executors.newCachedThreadPool(); executor.execute(new XXXRunnable);

executor.shutdown() // run to finish

Java语言有8种基本数据类型 – boolean, char, byte, short, int, long, float, double • 8种基本数据类型可分为4大类型

逻辑类型：boolean

字符类型：char

整数类型：byte, short, int, long

浮点类型：float, double

(1) byte型 – 关键字：byte；内存：1个字节，8位；取值范围：-2^7~2^7-1

(2) short型 – 关键字：short；内存：2个字节，16位；取值范围：-2^15~ 2^15-1

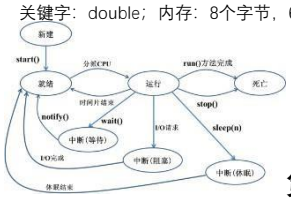
(3) int型 – 关键字：int；内存：4个字节，32位；取值范围：-2^31~2^31-1， 即-2147483648~2147483647，不能用来表示整个地球的居住人口

(4) long型 – 关键字：long；内存：8个字节，64位；取值范围：-2^63~2^63-1

字符型关键字：char；内存：2个字节，16位；最高位不是符号位，没有负数 取值范围：0~2^16-1， 即0~65535

关键字：float；内存：4个字节，32位 – 取值范围：10^-38~10^38和-10^38~-10^-38

关键字：double；内存：8个字节，64位 – 取值范围：10^-308~10^308 和 -10^308~-10^-30



第八章

JFrame JApplet JDialog 默认是 BorderLayout。JPanel 默认是 FlowLayout

JButton JTextField JTextArea JPanel JCheckBox JRadioButton JLabel JComboBox---JComponent LayoutManager: FlowLayout, BorderLayout, GridLayout, CardLayout, BoxLayout, Null

添加子组件 add()，删除组件 remove(comp)，删除全部 removeAll()

添加和删除后，要 validate()一下

JFrame setBounds(x,y,w,h) setSize(w,h) setVisible(true)

窗口默认不可见，要 setVisible

JScrollPane(comp)封装滚动窗格

JSplitPane(int how_to_split, comp_b, comp_c)拆分窗格，how_to_split 为HORIZONTAL_SPLIT、VERTICAL_SPLIT JSplitPane.setDividerLocation(double pos)

JDialog 有modaless 和modal（"有模"）， m od al 的会堵塞线程。 Dialog = new JDialog(jframe, str, has_model) If(dialog.getMessage() == MyDialog.YES) {……}

JinternalFrame-内部窗口 MDI

布局：

FlowLayout、BorderLayout、CardLayout、Gridlayout(m,n)、BoxLayout comp.setLayout(layout); layout.setHgap layout.setVgap

JFrame 布局：

win.add(bNorth, BorderLayout.NORTH); win.add(bSouth, BorderLayout.SOUTH); win.add(bEast, BorderLayout.EAST); win.add(bWest, BorderLayout.WEST); win.add(bCenter, BorderLayout.CENTER); win.validate();

setBounds(100, 100, 500, 300);

setVisible(true); setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

JTextField：

setText、getText、setEditable

JPasswordField： setEchoChar(char c)设置回显 char[] getPassword() 获得密码

actionEvent：

事件源、监视/监听器和处理事件的接口

class PoliceStation implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

//comp = (JTextField) e.getSource() String str = e.getActionCommand(); System.out.println(str); System.out.println(str.length());

}

}

JTextArea： setLineWrap(true) setWrapStyleWord(true)

append(s) insert(s, int x) getCaretPosition()获取光标位置 copy() cut()复制或剪切到剪切板 paste()粘贴

WindowListener 接口，WindowAdaptor 类最小化最大化之类的事件 addWindowListener

URL：

try

{

url=new URL("http://yahoo.com.cn");

}

catch(MalformedURLException e)

{

System.out.println("Bad URL:"+url);

}

InputStream URL.openStream()

可以用JEditorPane(String url)来显示网页

JEditorPane 对象调用 addHyperlinkListener(HyperlinkListener listener)获得监视器。

监视器需要实现 HyperlinkListener 接口，该接口中的方法是 void hyperlinkUpdate(HyperlinkEvent e)

UDP 发送：

byte data[ ]="近来好吗".getByte();

InetAddress address = InetAddtress.getName("www.sina.com.cn");

DatagramPacket data_pack = new DatagramPacket(data, data.length, address, 5678);

DatagramSocket mail_out = new DatagramSocket();

mail_out.send(data_pack);

用关键字**private修饰的成员变量和方法被称为私有成员变量和私有方法**。只有在本类中创建该类的对象时，这 个对象才能访问自己的私有成员变量和类中的私有方法。

用**public修饰的成员变量和方法被称为共有成员变量和共有方法**。当我们在任何一个类中用类A创建了一个对象a后，该对象a能访问自 己的public成员变量和类中的public方法。

不用private, public, protected修饰的成员变量和方法被称为友好成员 变量和友好方法。

用protected修饰的成员变量和方法被称为受保护的成员变量和受保护 的方法（即使在不同的包中，子类也能访问）。

.public类与友好类 • 类声明时，如果关键字class前面加上public关键字，就称这样的类是 一个public类。 **• 不能用protected修饰外部类，只能修饰内部类**（这种情况很少用）。 **• 不能用private修饰外部类，只能修饰内部类**（这种情况很少用）。 **• 不能用static修饰外部类，只能修饰内部类**（这种情况很少用）。

在类A中，可以访问对象a的以下成员 – **private, friendly (or default), protected, public** • 在与类A同package的另外一个类B中，可以访问对象a的以下成员 – **friendly (or default), protected, public** • 在类A的子类B中(不同package)，可以访问对象a的以下成员 – **protected, public** • 在与类A不同package的另外一个类C中，可以访问对象a的以下成员 – **public**