Java 语言概述 java 的特点：简单易学、平台无关性、面向对象、多线程、可移植、健壮、安全、动态 Java SE 桌面开发和低端商务应用的解决方案。 Java EE 是以企业为环境而开发应用程序的解决方案。 Java ME 是致力于消费产品和嵌入式设备的最佳解决方案。 JVM Java 虚拟机 JDK 开发工具包 JRE Java 运行环境 IDE 集成开发环境 .java 文件是源文件，通过 javac 命令编译后生成 .class 文件；.class 文件是字码结文件，即.java 文件编译后的代码。 标识符与基本数据类型及数组 数组拷贝是引用 arraycopy 方法 arraycopy(Object src,int srcPos,Object dest,int destPos,int length)运算符表达式控制语句 类与对象 封装继承多态 static 静态变量是和该类所创建的所有对象相关联的变量，改变其中一个对象的这个静态变量就同时改变了其它对象的这个静态变量。 静态方法（static method）只能调用该类的静态方法，不能调用实例方法。 实例方法可以操作实例变量、静态变量 静态方法只能操作静态变量，不能操作实例变量 实例方法必须通过对象来调用 静态方法可以通过类名调用 this 不可以出现在静态方法 import 语句必须写在 package 语句和源文件中类的定义之间 继承与接口及泛型 子类和父类在同一包中的继承性：子类自然地继承了父类中不是 private 的成员变量（即：friendly, protected, public）作为自己的成员变量，并且也自然地继承了父类中不是 private 的方法（即：friendly,protected, public）作为自己的方法。继承的成员变量和方法的访问权限保持不变。子类和父类不在同一包中的继承性：如果子类和父类不在同一个包中，那么子类只能继承父类的 protected,public 成员变量和方法，继承的成员变量和方法的访问权限保持不变。如果子类和父类不在同一个包里，子类不能继承父类的 friendly 成员变量和 friendly 方法。 方法重写：子类中定义一个方法，并且这个方法的名字、返回类型、参数个数和类型与从父类继承的方法完全相同。如果子类想使用被隐藏的方法，必须使用关键字 super super 关键字有两种用法：在子类中使用 super 调用父类的构造方法 在子类中使用 super 调用被子类隐藏的成员变量和方法 对象的上转型对象的实体是由子类负责创建的，只不过失掉了一些属性和功能而已。 abstract 类:• abstract 类不能用 new 运算符创建对象，由子类创建对象。• 若 abstract 类的类体中有 abstract 方法，只允许声明，不允许实现；• 该类的子类必须实现 abstract 方法，即重写（override）父类的 abstract 方法。• 一个 abstract 类只关心子类是否具有某种功能，不关心功能的具体实现。 接口：接口中的方法默认是 public 和 abstract 的 抽象类与接口的比较：抽象类中可以有 abstract 方法、非 abstract 方法；接口中只可以有 abstract 方法 •抽象类中可以有常量、变量；接口中只可以有常量 匿名类：匿名类可以访问外嵌类中的成员变量和方法 – 匿名类不可以声明静态成员变量和静态方法 异常类：继承 Exception 泛型类：使用泛型类声明变量、创建对象时，必须要指定类中使用的泛型的实际类型。 常见类集合类及泛型类应用 Date 类：System 类的静态方法 public long currentTimeMillis()获取系统当前时间 SimpleDateFormat(String pattern) 常用时间元字符 Calendar 类：getInstance()可以初始化一个日历对象 Math BigInterger NumberFormat LinkedList HashSet HashMap（Collection<Integer> collection = map.values(); Iterator<Integer> iter = collection.iterator();） TreeSet(升序排序 compareTo 方法 或者类的 compare 方法） TreeMap Stack 字符串及其应用 字符串构造 String (char a[]) String(char a[], int startIndex, int count) String 类提供了将字符串存放到数组中的方法：public void getChars(int start, int end, char c[], int offset ) String 类的实体不可修改，StringBuffer 类可修改 构造：StringBuffer() StringBuffer(int size) StringBuffer(String s) append 方法 char charAt(int index)：void setCharAt(int index, char ch)：StringBuffer insert(int index, String str) public StringBuffer reverse() StringBuffer delete(int startIndex, int endIndex) StringBuffer replace(int startIndex, int endIndex, String str) StringBuffer 与 StringBuilder – 功能几乎完全相同 – StringBuffer 是线程安全的，StringBuilder 不是线程安全的 – 如果字符串缓冲区被单个线程使用（这种情况很普遍），建议优先采用 StringBuilder，因为效率高 – 如果需要多线程同步，则建议使用 StringBuffer StringTokenizer 类： – StringTokenizer(String s)：为字符串 s 构造一个分析器。使用默认的分隔符集合，即空格符（多个空格被看做一个空格）、换行符'\n'、回车符'\r'、tab 符'\t'、进纸符'\f' – 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 （Pattern p; Matcher m; String input = "0A1A2A3A4A5A6A7A8A9"; p = Pattern.compile("\\dA\\d"); m = p.matcher(input);）正则表达式：\d 0~9 的任何一个数字;\D 任何一个非数字字符;\s 空格类字符，'\t', '\n', '\x0B', '\f', '\r' ;\S 非空格类字符;\w 可用于标识符的字符（不包括美元符号）;\W 不能用于标识符的字符;? 0 次或 1 次; * 0 次或多次;+ 1 次或多次;{n} 恰好出现 n 次;{n,}至少出现 n 次;{n, m} 出现 n 次至 m 次 正则表达式匹配 while(m.find()) str = m.group() 字符串的替换 public String replaceAll(String regex, String replacement) 字符串的分解 public String[] split(String regex) 线程 • 程序是一段静态的代码，它是应用软件执行的蓝本。• 进程是程序的一次动态执行过程，它对应了从代码加载、执行至执行完毕的一个完整过程，这个过程也是进程本身从产生、发展至消亡的过程。• 线程是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，形成多条执行线索，每条线索，即每个线程也有它自身的产生、存在和消亡的过程，也是一个动态的概念。 线程的 4 种状态：新建、运行、中断、死亡 线程优先级：setPriority(int grade) 1-10 Runnable 接口 （String s1="treasurer"; // 会计 String s2="cashier"; // 出纳 Bank bank = new Bank(s1,s2);Thread zhang;Thread cheng;zhang = new Thread(bank); // 目标对象 bank cheng = new Thread(bank); // 目标对象 bank zhang.setName(s1); cheng.setName(s2); bank.setMoney(120); zhang.start(); cheng.start();） （public void run(){while(true) {money = money - 10; if(Thread.currentThread(). getName().equals(name1)){System.out.println(name1 + ": " + money); if(money<=100) {System.out. println (name1 + ": Finished"); return; } }else if(Thread.currentThread(). getName().equals(name2)){ System.out.println(name2 + ": " + money); if(money<=60){ System.out.println(name2 + ": Finished"); return; } } try{ Thread.sleep(800); } catch(InterruptedException e) {}} } }）1. Thread 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 接口的类的实例，该实例对象称作所创建线程的目标对象，2.关于 run()方法中的局部变量 • 对于具有相同目标对象的线程，当其中一个线程享用 CPU 资源时，目标对象自动调用接口中的 run()方法，当轮到另一个线程享用 CPU 资源时，目标对象会再次调用接口中的 run()方法。不同线程的 run()方法中的局部变量互不干扰，一个线程改变了自己的 run()方法中局部变量的值不会影响其他线程的 run()方法中的局部变量。线程的常用方法：start、run、sleep、isAlive、currentThread、interrupt • 线程同步是指多个

线程要执行一个 synchronized 修饰的方法，如果一个线程 A 在占有 CPU 资源期间，使得 synchronized 方法被调用执行，那么在该 synchronized 方法返回之前（即 synchronized 方法调用执行完毕之前），其他占有 CPU 资源的线程一旦调用这个 synchronized 方法就会引起堵塞，堵塞的线程要一直等到堵塞的原因消除（ 即 synchronized 方法返回），再排队等待 CPU 资源，以便使用这个同步方法。使用 wait(),notify(),notifyAll()协调同步线程：使用 wait()方法可以中断方法的执行，使本线程等待用 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：一个线程 A 在占有 CPU 资源期间，可以让线程 B 调用 join()方法和本线程联合，如：B.join(); 我们称 A 在运行期间联合了 B。•如果线程 A 在占有 CPU 资源期间一旦联合 B 线程，那么 A 线程将立刻中断执行 守护线程：一个线程调用 void setDaemon(boolean on)方法可以将自己设置成一个守护（daemon）线程，例如：thread.setDaemon(true);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。文件及输入输出流 • 9.1 文件 构造方法 文件的属性 目录 文件的创建与删除 运行可执行文（Runtime ec = Runtime.getRuntime(); File file = new File("C:\\windows", "Notepad.exe"); ec.exec(file.getAbsolutePath());) • 9.12 使用 Scanner 解析文件 创建 Scanner 对象，并指向要解析的文件 （File file = new File("hello.java"); Scanner scanner = new Scanner(file);）使用 useDelimiter 方法指定正则表达式作为分隔标记(while(scanner.hasNextDouble()){ fare = scanner.nextDouble(); sum = sum+fare;}) • 9.3 文件字符流 FileReader 类 – int read() - int read(char b[ ]) – int read(char b[ ], int off, int len)：读取 len 个字符并存放到字符数组 b 中，返回实际读取的字符数目；如果到达文件的末尾，则返回 - 1。其中，off 参数指定 read 方法从字符数组 b 中的什么地方存放数据 FileWriter 类 public void write(char b[]) -public void write(char b[], int off, int len) -void write(String str)- void write(String str, int off, int len) • 9.5 缓冲流 然后，input 调用 readLine()顺序读取文件的一行。（FileReader fr = new FileReader("Student.txt"); BufferedReader input = new BufferedReader(fr); ）;write(String s) • 9.2 文件字节流 FileInputStream 类 int read(byte b[ ]); FileOutputStream 类 void write(byte b[]) • 9.8 数据流 DataInputStream 类和 DataOutputStream 类 DataInputStream(InputStream is) • DataOutputStream(OutputStream os) • 9.9 对象流 ObjectInputStream 类和 ObjectOutputStream 类 • 9.10 序列化和对象克隆 使用对象流很容易获取一个序列化对象的深度克隆（原对象有引用型变量的时候）•我们只需将该对象写入到对象输出流，后用对象输入流读回的对象就是原对象的一个深度克隆。• 9.11 随机读写流 RandomAccessFile(String name, String mode)• 9.13 文件锁 JDK1.4 增加了一个 FileLock 类，该类的对象称做文件锁。RondomAccessFile 创建的流在读写文件时可以使用文件锁，那么只要不解除该锁，其它线程无法操作被锁定的文件。1. RandomAccessFile input = new RandomAccessFile ("Example.java", "rw"); 2. FileChannel channel = input.getChannel(); 3. FileLock lock = channel.tryLock(); • 9.6 数组流 字节输入流：ByteArrayInputStream(byte[] buf, int offset, int length)字节输出流：ByteArrayOutputStream(int size) public byte[] toByteArray()：返回输出流写入到缓冲区的全部字节 • 9.7 字符串流 构造方法：public StringReader(String s) public int read(char[] buf, int off, int len) StringWritter(int size); public void write(String str, int off, int len) GUI 编程 10.1 AWT 组件与 Swing 组件概述 AWT 是 Abstract Window Toolkit（抽象窗口工具包）的缩写。Java 1.2 中的 javax.swing 包提供了 Swing 组件，大部分组件是轻组件，没有同位体，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。• 10.2 JFrame 窗体 • 10.5 中间容器 JPanel 面板 JScrollPane 滚动窗格 JSplitPane 拆分窗格 JLayeredPane 分层窗格 • 10.23 对话框 JDialog 类 对话框分为无模式（modeless）和有模式（modal）两种。•无模式对话框处于激活状态时，程序仍能激活它所依赖的窗口或组件，它也不堵塞线程的执行。•有模式对话框处于激活状态时，只让程序响应对话框内部的事件，程序不能再激活它所依赖的窗口或组件，而且它将堵塞当前线程的执行，直到该对话框消失不可见。JOptionPane 类：输入对话框 showInputDialog 消息对话框 showMessageDialog 确认对话框 showConfirmDialog 颜色对话框 ColorshowDialog 文件对话框 showOpenDialog 、showSaveDialog• 10.3 菜单组件 JMenuBar 菜单条 JMenu 菜单 JMenuItem 菜单项 JTextField 文本框 JPasswordField 密码框 发生 ActionEvent 事件的事件源对象获得监视器方法是：– addActionListener(ActioListener listener); JTextArea 文本区 • 10.4 布局设计 FlowLayout（水平垂直）、BorderLayout（默认，东西南北）、CardLayout（选项卡）、GridLayout（网格区域）、BoxLayout（盒式布局）• 10.14 窗口事件 WindowListener 接口 WindowAdapter 适配器 • 10.15 鼠标事件 MouseListener 接口与 MouseMotionListener 接口 网络编程 • 11.1 URL 类 一个 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 JVM 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 public URL(String spec) throws MalformedURLException• 11.2 读取 URL 中的资源 URL 对象调用 InputStream openStream() • 该方法可以返回一个输入流，该输入流指向 URL 对象所包含的资源。通过该输入流可以将服务器上的资源信息读入到客户端。• 11.3 显示 URL 资源中的 HTML 文件 JEditorPane 类 • 11.4 处理超链接 JEditorPane 对象调 addHyperlinkListener(HyperlinkListener listener)获得监视器。• 监视器需实现 HyperlinkListener 接口，该接口中的方法是 void hyperlinkUpdate(HyperlinkEvent e) • 11.5 InetAddress 类 获取 Internet 上主机的地址• InetAddress 类的静态方法：getByName(String s); • 11.6 套接字 Socket 套接字连接就是客户端的套接字对象和服务器端的套接字对象通 过输入、输出流连接在一起 (1) 服务器建立 ServerSocket 对象 (2) 客户端创建 Socket 对象 (3) 流连接 • 11.7 使用多线程处理套接字连接 • 11.8 UDP 数据报 • 基于 UDP 通信的基本模式是 • (1) 将数据打包，称为数据包（好比将信件装入信封一样），然后将数据包发往目的地。 • (2) 接收别人发来的数据包（好比接收信封一样），然后查看数据包中的内容。byte data[ ]="近来好吗".getByte(); InetAddtress address = InetAddtress.getName ("www.sian.com.cn"); DatagramPacket data_pack = new DatagramPacket(data, data.length, address, 5678); (2)发送数据 • 11.9 广播数据报 1.设置组播地址 2.创建多点广播套接字 3.设置广播的范围 4.加入组播组 5.广播数据和接收数据 • 11.10 Java 远程调用