第一章:Java 运行平台： JDK 包含 JRE 包含 JVM JavaSE(J2SE)标准版平台（桌面开发和低端商务应用、 Applet)； Java EE （企业） API、开发网络、 web 应用 JavaEE(J2EE)企业版平台、 JavaME(J2ME)(嵌入 式设备) （微型） 移动设备； Java CARD SIM、ATM 卡。 javac.exe 编译器 java.exe 解释器 jdb.exe 调试器 javap.exe 反编译。 Java 特点： 简单、 面向对象、分布式、解释型、健壮、安全、架构中立、 可移植、高性能、多线程、动态。 描述 Java 技术的主要特性:java 虚拟机,垃圾回收,代码安全性。 Java 虚拟机： 字节代码， 垃圾回收有一定滞后性.命令行编译： javac Hello.java→编译生成 Hello.class →运行 java Hello。PATH： 操作系统运行环境的路径， CLASSPATH： JAVA 运行应用程序时所需要的类包的路径

第二章 1.关键字:transient、strictfp、volatile、native 2.数据范围:077 0x3ABC 3. byte 运算必须转化 byte s = byte（a+b） 4.byte short int long 字节 1248,范围-2^(n-1)~ 2^(n-1)-1 5.boolean 单个 4 字 节 、数组中 1 字节 6.char 2 字节 0~65535(0~2"16-1)'\u????'(四位十六进制)7.float4 字节 1.4E-45 ~ 3.4E+38(f) 9.double8 字节 4.9E-324~1.7E308 7. int [][]a=new int[10][]; 下标越界 ArrayIndexOutOfBoundsException 8.Instanceof 确定某个对象是否属于某个类 9.>>右移补符号>>>补 0 10.switch char byte short int String； 10.引用型变量的引用地址和基本类型变量存在栈中，对象数 据存在堆中。 11. char 与 int 相加结果是 int 12.jar cfm dest.jar hello.mf a.java b.java

第三章:1.面向对象特性:封装继承多态 3.没有实体的对象使用会有 NullPointerException 5.重载:参数列表不同即可(返回值可以不同) 6..java.lang.Number 中有基本类型的类包装7.面向对象好处：模块化、信息隐藏、代码重用、易调试 8.用 static 修饰的成员方法称为静态方法（类方法），不使用则称为实例方法；类方法可通过类名或对象名来调用 9.Package 放到 import 前 10.重载编译时处理重写运行 时处理. 10. abstract 类可有构造方法，非抽象类不能有抽象方法 11.子类不继承父的构造方法。 Super 调用父类构造方法必须放第一行 13.接口方法默认 abstract public 变量默认final 必须初始化 18. 非 static 内部类不可声明静态变量和静态方法 20. A.B obj = new A().new B(); 21 匿名类一定是内部类，可以继承，匿名类不可声明静态成员变量和静态方法。 22.异常类: Exception 的相应子类 23.上转型变 量:不能通过上转型对象变量访问子类对象实体中的成员变量和成员方法；通过上转型对象变量访问子类对象实体重写的父类成员方法，执行的代码是子类重写的方法体；可通过上转型对象变量访问父 类被隐藏的成员变量； 可通过强制类型转换将上转型对象变 量转换为子类对象变量 1.类方法中不能引用对象变量不能调用类的对象方法不能调用 superthis 关键字； 4.类方法不能被重写。跨包继承只 能继承 protected 和 public 方法，作为成员变量不能挎包访问 protect 方法。 5.必须在 try-catch 中抛异常 6.泛型不能为基本数据类型，创建声明对象必须指定实际类型。 泛型只能调 object 类的方法。 同一个包另外的类可以访问除 private。子类可访问 protect 和 public。不同包只能访问 public。

第五章:String 表示一个 UTF-16 格式的 2byte 字符串 2. java.lang.String 包 4.String 的长度是 str.length() 数组元素个数是 arrey.length （无括号） 5.比较字符串字典序大小用 str.comparTo(String str) 返回 0 负数代表等于小于 7.str.equalsIgnoreCase(str2)忽略大小写 str.toUpperCase()str.toLowerCase() 字符串全字符转为大小写 6.str.startsWith(String string) .endsWith(String string) 7. str.contains(子串)判断有无子串 int indexOf(String s)判断字串第一 次出现的位置没有该串返回-1 8.String substring(int startpoint,int endpoint)截取从 stratpoint 到 endpoint-1 的子串 9. String replaceAll(String s1, String s2)将 str 中的所有 s1 子串改为 s2 10. String trim()返回去除前后空格的串 11.字符串转化为数据类型:int a=Integer.parseInt(String s, int radix) Integer a=Interger.valueOf(String s) 12.数据类型转化为字符串,String.valueOf(数据值) 或者 str = " "+数 或者 str = Integer.toString(int a)12.进制转换:Integer.toString(值,进制);返回一个值在该进制下的字符串 13. public void getChars(int start, int end, char c[], int offset)截 取 string 中 start 到 end 到 从 c 的 offset 位 置 开 始 （ 不 包 括 end ） 14. public char[] toCharArray() 字 符 串 转 字 符 数 组 15. 用 字 节 数 组 构 建 字 符 串 String(byte[]),String(byte[],start,length)str.getBytes 获得字符串字节表示 15.str.charAt(i)获得 i 位置字符 16.str.concat(String str2)字符串连接,返回一个新的字符串 17.str.replace(char a,char b)返回一个将 str 中 all 字符 a 换成 b 的串 18.StringBuilder 和 StringBuffer 构造函数 StringBuffer(int capacity)容量为 capacity ,StringBuffer()默认容量 16 StringBuffer(String s)容量为 len(s)+16 19. 常 用 方 法 append(num/char[]/String) delete(start,end)(删 除 start~end-1) deleteCharAt(index) insert(offset, 数 据 字 符 数 组 字 符 串) 字 符 数 组 有 insert(index,char[]str,start,len) replace(start,end,String) 指定位置替换字符串(即删除 start 到 end-1 的串并换成新串)setCharAt(index,char) reverse()反转并返回 StringBuilder、toString 返回 String 对象 19.String[] split(String regex) 用 regex 作为分隔符 20.Scanner(String str) hasNext ,Scanner.useDelimiter(regex)用正则表达式作为分隔符 "[^0123456789.]+"匹配非(数字和小数 点)String.format("%d",123); 21.线程安全 StringBuffer 22.StringBuffer 的 setLength(int len)，设置序列长度，大了就增加空字符，小了就截。 23. StringTokenizer nextToken() hasMoreToken。p = Pattern.compile("\\dA\\d");m = p.matcher(input);

第六章:泛型 java.util.*;1.Java.lang.Math 2.LinkedList add(E) add(index E) remove(index) indexOf(E) lastIndexOf(E) set(index, E) contains(O) isEmpty() size() toArray() get(index) 3.集合分为两大类,实现了 Collection 接 口 (List:ArrayList,LinkedList,Vector 和 Set:HashSet,TreeSet,LinkedHashSet) 和 实 现 了 Map 接 口 (HashMap,TreeMap,HashTable) 4.Vector 最 安 全 。 线 程 安 全 版 List list = Collections.synchronizedList(new ArrayList<>()/或者为 LinkedList<>()5.HashSet(不保证元素顺序不变化)如果是 E 是自定义对象,则需要在对象中重写 int hashCode{return 自定义 hash 编 码 ;}boolean equals(Object obj){if(obj.hashCode()==this.hashCode())return true;return false;} 线程安全 :Set set = Collections.synchronizedSet(new HashSet<>()); 操 作:add,remove,contains,size,clone 无 get,用迭代器 获得元素(Treesetis=new TreeSet();Itertator it=is.iterator()返回迭代器 while(it.hasNext){it.next();})还有集合操作:A.addAll(B)并 retainAll 交 removeAll()差 6.TreeSet 是有序的集合若 E 为自定义类,需要实现 Comparable<E>接口重写 ComparTo 方法 public int compareTo(E obj) { return this-obj}升序.标点<大写<小写。 线 程 同 步 SortedSet s = Collections.synchronizedSortedSet(new TreeSet<>());7.HashMap 无序 快 TreeMap 有序 慢 LinkedHashMap 有序快 Iterator<testClass> iter = a.iterator();7.HashMap 注 意若 T 为自定义类 , 要跟 HashSet<> 一样重写两个方法 操作 :put(Key,Value) get(Key) containsKey(Key) containsValue(Value) remove(Key) Set set = hashmap.keySet() Collection values = hashmap.values()可用迭代器 Iterator 或 ArrayList 包装强制转换 values 8.TreeMap 要 在自定义的 Key 类中实现 Comparable 接口 (同上) 或者在构造 TreeMap 时提供一个比较器对象参数 , 比较器 为实现泛型 接口 Comparator 的类 , 方法:public int compare(E o1,E o2) 线程安全： Map m=Collections.synchronizedMap(new HashMap(..)) 10.stack:push pop peek search()->1 List 可以调用 Collections.sort 进行排序

第七章:异常处理 java.lang.*;java.io.*; java.util.*;java.net.*： 2.异常-生成异常对象-交给 JVM(过程称为抛出异常)JVM 寻找(遍历栈)能处理该异常的方法,交给这个方法(捕获异常)(默 认处理是输出异常信息终止程序)3.标准异常类都是 Exception 的子类 4.Throwable(java.lang 包)直接子 类 :Error:OutOfMemoryError, ThreadDeath(致 命 错 误 , 与 JVM 相 关 , 由 系 统 处 理 入 线 程 死 亡 , 内 存 溢 出)Exception(异 常) 4.Exception(String)Exception() 5.异常的方法 printStackTrace()输出异常的信息 6. try- catch-finally 语句块 finally 块是个可选项(一定执行)7.常见异常： 运行异常 ArithmeticException 、ArrayIndexOutOfBoundsException 、 NullPointerException 、 ClassCastException(类型转换异常)NumberFormatException 非运行异常:IOException 、 FileNotFoundException、SQLException8.自定义异常类:继承 Exception,构造函数(Stringstr),并且要 super(str) 8.Exception 分为运行时异常 RuntimeException(不检查异常)和非运行时异常 (检查异常) 10.子类重写父类的带抛出异常的方法:1.不声明抛出异常 2 声明抛出的异常必须是其父类方法声明抛出的那种异常或者其子类异常 ·Throwable->Error 和 Exception

第八章:3.没有结束 run()方法之前不能再 start()否则 IllegalThreadStateException 4.run 用户程序不得调用只能系统调用 5.创建线程两种方法(可以同时用):1.继承 Thread 类(已经实现 Runnable 接口)重 写 run()方法 2.实现 Runnable 接口实现 run()方法 6.方法:name()start()run()setName()getName()toString()(名称,优先级,所属线程组)currentThread(当前运行的线程引用)isAlive()(start-run 为 true, 其余 为 false)sleep(int ms)(静态方法)interrupt()(中断休眠进入就绪)wait()(等待 notify())notify()(唤醒正在等待的线程)notifyAll(唤醒所有等待的线程) join()t.join()让 t 运行,t 运行完当前线程再继续(wait 和 sleep 和 join 放在 try-catch 中异常为 InterruptedException) 4.优先级(启动前设置)t.setPriority(0~10 值大级大)(超出范围 IllegalArgumenException)多个线程排队优先级高的先运行相同优先"先到先 得",getPriority() 5.synchronized 关键字(常在方法中 synchronized(args){语句})方法声明 synchronized(与 static 同位置)6.协作:wait()暂停线程执行并释放锁,notify() 会唤醒一个等待的线程(两线程协作,常 需要轮询等待 while(balance>=5000){try{wait()}catch(InterruptedException)}})balance 是一个必要的控制元素。进行一系列操作后再 notify()(注 意,有 notify()的方法必须是同步的)7.join 挂起:B.start()等 到 B 运行就把 B 加入 while(threadB.isAlive()==false){} try{ threadB.join(); } catch(InterruptedException e){} 7. 具有相同优先级的 多个线程的调度不一定是分时的 9. 如有高优先级的线程就绪，正在运 行的低优先级的线程将暂停执行 10.多个线程的运行顺序其实与线程的优先级有关，与线程的启动顺序无关 10. 一个已经运行的线程在没有进入死亡状态时，不能再给线程分配实体 wait(), notify()和 notifyAll()不允许重写 。IllegalMonitorStateException 对象监视器和 wait 必须作用于同一个线程

第九章: IO import java.io.*; import java.util.*;(Scanner 类) Scanne(file)文件流注意 FileNotFoundException 读取文件注意 IOException 2.File 类文件对象并不涉及对文件的读写操作3.File 类 :File("pathname") getName canRead canWrite exists length getAbsolutePath getParent isFile isDirectroy mkdir 创建目录 list 返回目录下的文件名数组,listFiles 文件对象数组, file.delete()删除文件 renameTo(File newName) long lastModified() 文件过滤需实现 FilenameFilter 接口并重写 boolean accept(File dir, String name) 4.字符流 :FileReader:read()读取单个字符 int read(char[])(需要强制转化 char) int read(char[],start,len)(返 回读取的个数,无法读取返回-1) close() FileWriter: write(int c) write(char[],start,len) write(String)flush()close() 5.FileReader(String/File)FileWriter(String/File)FileReader fr = new FileReader("out.txt");int c;char[] buffer = new char [message.length()] ; while((c=fr.read(buffer,0,buffer.length))!=-1) {String str = new String(buffer,0,c);System.out.println(str);} 记 得 close() 6. 缓 冲
流:BufferedReader(Reader)BufferedWriter(Writer)常用 FileReader 和 FileWriter 对象为参数构建 br.readLine() (while((str=br.readLine())!=null && str.length()!=0)){ bw.write(String) bw.newLine()}bw.flush(), bw.close() 7.字节流 FileInputStream 类(以字节为基本处理单位):read()(读一个字节)read(byte[])(读取一定量的字节)read(byte[],start,len)(读取 len 个字节入 byte)read 到达结尾返回(-1) available()返回可读 取字节数 close()关闭 8.FileOutputStream:write(byte[]),write(byte[].start,len) flush()刷新输出流并强制写出所有缓存的输出字节 close()关闭
9.FileInputStream(String)/(File)(FileNotFoundException)通常是 byte buffer[] = new byte[len];while(in.read(buffer,0,len)!=-1){String str = new String (byte,0,len) System.out.print(str);}in.close();catch..10.FileOutputStream(文件夹 FileNotFoundException 不存在自动创建)常 int count,n=512;byte buffer[]=new byte[n];count=System.in.read(buffer); FileOutputStream out=newFileOutputSteam(filename); out.write(byte,0,count); out.close(); 11.DataInputStream(FileInputStream)DataOutPutStream(同上) DataInputStream(new FileInputStream(File/Name))方法:readInt,readChar,readLine,readShort,readDouble,readFloat. DataOutputStream: write 数据类型 writeChars()输出字符串. 12. 实现 implements Serializable 接口进行序列化 Goods TV2 = (Goods)objectIn.readObject() objectOut.writeObject(TV1); InputMismatchException 序列化拷贝 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(); 13.数组字节流不发生异常， 数组字符流可能发生 IOException 14.文件锁 FileChannel channel = input.getChannel(); FileLock lock = channel.tryLock(); lock.release(); 15. InputStream 字节流顶层父抽象类 Reader 字符流顶层父抽象类

第十章 javax.swing.*; class FileWindow extends JFrame implements ActionListener 1.每次添加新组件，容器调用 validate() JFrame, JApplet, JDialog 都是重组件 JComponent 类的子类都是轻组件 所有的 轻组件都是容器 setBounds(int a, int b, int width, int height)： setSize(int width, int height) setVisible(boolean b) 2.顶层容器:JFrame,JDialog,Japplet 3.中间层容器:Jpanel 默认 FlowLayout 布局 JDialog 必须依赖于窗口或组件 4.JFrame 默认 BoarderLayout 布局 FlowLayout 元素可以被移动 BorderLayout 只五个区域 选项卡窗格的默认布局是 CardLayout 布局 可以容纳多个组件 GridLayout 网格布局 BoxLayout createHorizontalStrut(int width) createVertialStrut(int height) 5.窗口只能有一个菜单条 JMenuItem 菜单项是 JMenu 的父类 JTextField 单行文本 setText 调用 addActionListener 添加监视器实现 addActionListener 接口的 actionPerformed 方法 可以利用接口回调或者直接调用 this 6. JTextArea 多行文本 append(String s)尾加文本 DocumentEvent 事件 DocumentListener 接口三个方法都必须被 实现 changedUpdate remove insert getDocument() setOpaque 设置透明 public Point getLocation() getBounds()：当撤销窗口图标化时， 监视器调用 WindowDeiconified()方法后还会调用 WindowActivated() 方法 7.setDefaultCloseOperation JFrame.DISPOSE_ON_CLOSE JFrame.EXIT_ON_CLOSE JFrame.DO_NOTHING_ON_CLOSE JFrame.HIDE_ON_CLOSE 8. FocusEvent FocusListener 接口实现 focusGained 和 focusLost 方法 9. KeyEvent KeyListener 接口实现 keyPressed keyTyped KeyReleased 10.Timer(int a, Object b)创建一个计时器每隔 ams 回调 ActionListener 中 actionPerformed 方法

第十一章: java.net.*;: 1.URL 三部分:协议地址资源:协议 http、ftp、file 协议 URL MalformedURLException getProtocol,Host,Port,Path.Query,Ref 获得协议,主机,端口,路径,查询,URL 对象的引用 从 url 里获取资源 InputStream in = url.openStream(); 新建一个线程以避免阻塞主线程 2.JEditorPane 类可以解释执行 html 文件抛 IOException setPage 3.JEditorPane 显示新页面对象调用 addHyperlinkListener(HyperlinkListener listener)获得监视器, 监视器需实现 HyperlinkListener 接口中 void hyperlinkUpdate(HyperlinkEvent e)方法。 4. InetAddress 对象含 Internet 主机地址的域名和 IP 地址 InetAddress address_1 = InetAddress.getByName("www.sina.com.cn"); getHostName() getHostAddress() (UnknownHostException) 5.TCP 套接字面向连接:端口号与 IP 地址的组合得出一个网络套接字 16 位端口(0~65535)(0~1023 被占)服务器端建立 ServerSocket 对象 ServerSocket waitSocketConnection = new ServerSocket(1880); 使用方法 accept()接收客户端的套接字连接请求, Socket socketAtServer = waitSocketConnection.accept(); 客户端建立 Socket 对象 Socket socketAtClient = new Socket(localhost, 1880);或调用 connect(InetSocketAddress endpoint)方法建立连接 完成建立 Socket 对象后进行 IO 流连接。通过 DataInputStream 建立输入输出 socket.getInputStream() 然后利用 IO 完成信息传递 6.UDP 无连接不可靠 DatagramPacket =new DatagramPacket (str.getBytes(),str.length(),InetAddress.getByName("…"),8000); 发送 DatagramSocket.send(DatagramPacket); 接收 UDP: dgs = new DatagramSocket(8000); byte[] b = new byte[1024]; dgp = new DatagramPacket(b,b.length); dgs.receive(dgp); String message = new String(dgp.getData(), 0, dgp.getLength());发送 ClientSocket.send(Packet(参数多 package))接受 ServerSocket.receive(参数的 package) 发送或者接收完之后要 Socket.close(); 7.TCP 面向连接,先建立连接再通信 Socket("域名如 time.nist.gov","端口").ServerSocket(端口)ServerSocket.accept()返回一个 Socket|(Server)Socket.getInputStream 通常包装为 Scanner(InputStream)或者 StringReader(new InputStreamReader(InputStream))用于接收信息|(Server) Socket.getOutPutStream 通常包装为 PrintStream(OutputStream), print(Obj) flush() close() 还有 PrintWriter(OutputStream,true) println(String) flush() close()(注意 IOException)半关闭:shutdownOutput()可以关闭输出流,只接收。多线程:while(true)去接受 socket,开启一个 Handler(实现了 Runnable 接口)线程去处理。注意流的关闭与 socket 的关闭。TCP 连接实现服务器 InetAddress address = InetAddress.getByName("127.0.0.1"); DatagramPacket data = new DatagramPacket(b,b.length,address,4000); DatagramSocket mail = new DatagramSocket();mail.send(data); 7.广播数据报 当 a 小于 128,那么 b.c.d 就用来表示主机, A 类地址。当 a 大于等于 128 并且小于 192,a.b 表示网络地址, c.d 表示主机地址, B 类地址。当 a 大于等于 192,则网络地址是 a.b.c,d 表示主机地址, C 类地址。 224.0.0.0 与 239.255.255.255 之间 D 类地址, 不代表特定主机的位置。 ABC 类必须加入 D 类。使用 InetAddress 类创建组播地址 InetAddress group = InetAddress.getByName("239.255.8.0"); 创建多点广播套接字 public MulticastSocket(int port) throws IOException 设置广播的范围 MulticastSocket. setTimeToLive(int ttl) 加入广播组 MulticastSocket. joinGroup(InetAddress mcastaddr) 发送接受同 TCP。

String Integer.parseInt(String) String.valueOf(int)
length() equals(String s) startsWith(String s) endsWith(String s) compareTo(String s) indexOf(String s) charAt(int index)
String replaceAll(String s1, String s2) String substring(int startPoint) String trim() char[] toCharArray() byte[] getBytes()

线程 sleep(int millsecond) isAlive() Thread.currentThread() interrupt() synchronized() wait() notify() notifyAll()

GUI setVisible(boolean b) JPanel面板的默认布局是FlowLayout; BorderLayout; CardLayout; .GridLayout; BoxLayout;
JButton按钮 JCheckBox复选框 JRadioButton单选按钮 JLabel标签 JComboBox下拉列表 JTable表格 JTree树 JProgressBar 进度条
WindowListener; MouseListener; MouseMotionListener; KeyListener; ActionListener

接口 interface; implements; 常量为public static final, 方法为public abstract; 在实现接口中的方法时, 一定要用public来修饰

Scanner useDelimiter(正则)分隔标记;
模式匹配 Matcher find() matches() lookingAt() String replaceAll(String) String replaceFirst(String)
Pattern p = Pattern.compile("A\\d"); Matcher m = p.matcher(input);
while(m.find()) { String str = m.group(); System.out.println(str); }

IO流、文件
– 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 file = new File("C:\\windows", "Notepad.exe"); Scanner scanner = new Scanner(file);
FileReader fr = new FileReader("Student.txt"); BufferedReader input = new BufferedReader(fr); while((s = input.readLine())!=null)
FileWriter fw = new FileWriter("hello.txt"); BufferedWriter output = new BufferedWriter(fw); output.write(String s)
DataInputStream(InputStream is) DataOutputStream(OutputStream os) readUTF() writeUTF(String)

TCP套接字
Server:
```
ServerSocket server; Socket socketAtServer;
try { server = new ServerSocket(8888); socketAtServer = server.accept();
    DataInputStream input = new DataInputStream(socketAtServer.getInputStream());
    DataOutputStream output = new DataOutputStream(socketAtServer.getOutputStream());
    while(true) { Scanner scanner = new Scanner(System.in); String str = scanner.nextLine(); output.writeUTF(str); }
} catch(IOException e) { System.out.println(""+e); }
```
Client:
```
Socket socketAtClient;
try { socketAtClient = new Socket("localhost", 8888);
    DataInputStream input = new DataInputStream(socketAtClient.getInputStream());
    DataOutputStream output = new DataOutputStream(socketAtClient.getOutputStream());
    while(true) { String str = null; while (str == null) { str = input.readUTF(); } System.out.println(str); }
} catch(IOException e) { System.out.println("Unable to connect to the server"); }
```

UDP数据报
```
DatagramPacket pack1, pack2; DatagramSocket mail1, mail2; InetAddress address; byte[] b1 = new byte[8192], b2;
try { pack1 = new DatagramPacket(b1,b1.length); address = InetAddress.getByName("localHost");
    mail1 = new DatagramSocket(8886); mail2 = new DatagramSocket();
    while(true) { mail1.receive(pack1); String message = new String(pack1.getData(),0,pack1.getLength());
        b2 = message.getBytes(); pack2 = new DatagramPacket(b2,b2.length,address,8889); mail2.send(pack2); }
} catch(Exception ignored) {}
```

广播数据报 InetAddress group = InetAddress.getByName("239.255.8.0"); public MulticastSocket(int port) throws IOException
public void joinGroup(InetAddress mcastaddr) throws IOException leaveGroup(InetAddress mcastaddr) throws IOException
public void send(DatagramPacket p) throws IOException public void receive(DatagramPacket p) throws IOException

元字符	在正则表达式中的写法	意义
.	.	代表任何一个字符
\d	\\d	代表0~9的任何一个数字
\D	\\D	代表任何一个非数字字符
\s	\\s	代表空格类字符，'\t', '\n', '\x0B', '\f', '\r'
\S	\\S	代表非空格类字符
\w	\\w	代表可用于标识符的字符（不包括美元符号）
\W	\\W	代表不能用于标识符的字符

带限定符号的模式	意义
X?	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恰好出现n次
X{n,}	X至少出现n次
X{n, m}	X出现n次至m次

\\p{Punct}表示标点符号 !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
public String replaceAll(String regex, String replacement)
pattern=pattern1|pattern2;（或） [a-z&&[def]]: 代表d, e或f中的任何一个（交）
– [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