基本概念 (安全性、可移植性、平台无关性) 注意 try 语块的使用

（0.5）开发运行 JAVA 的三个步骤：编译源程序、编译生成字节码、解释运行字节码（1）2018 年出版 JAVA SE 10、11（2）三个平台：J2SE：桌面应用/服务器，J2EE：企业级，J2ME：嵌入式（3）JAVA 能够识别的二进制代码是字节码（4）没有 unsigned 类型（5）short 和 char 两个字节，char：0~65535，char 不能转成 short

（6）float：+/-10^(+/-38)，最高只有 8 位有效数字，格式：1.0 f（7）double：+/-10^(+/-308)，最高只有 17 位有效数字

（7）%f 最多只能能保留 6 位小数（9）boolean b = a instanceof Class 判断 a 是否为 Class 对象（10）switch 中表达式的值只能是 int 或 char，且 case 不能重复（11）类名规范：所有字母开头大写，与方法名规范不同（12）运行命令：java、编译命令：javac (-d)（13）java 类库存在于 jdk18.0_144\jre\lib\rt.jar（14）protected 和 private 不能修饰外部类和接口，static 不能修饰外部类

基本数据类型的封装（位于 java.lang 中，无需导入）

（15）都有默认装箱方法和显示装箱方法 valueOf(基本型/String) （16）String -> 基本型：封装类.parseXxx (String)

（17）都有默认拆箱方法和显示拆箱方法 xxxValue( ) 从对象转为基本型（18）都有静态方法 toString(对象/基本型)

（19）相互间用 compare、compareTo 或 equals 比较

（20）String 类：位于 java.lang 中，Str -> 基本型：封装类.parseXxx (String)、基本型->Str: String.valueOf(基本型)、Str->char[]: .toCharArray( ) 或者 getChars(start, end, char[], offset) 、只能用 equals(), compareTo() 、starts/ens-With(Str)、.indexOf(substr)匹配子串位置、substring(startPoint)、trim()、replaceAll(s1,s2)、split(Str)返回 Str 数组、replace("正则"，String)

（21） StringBuilder/Buffer 类：构造(String)，方法：append(String)、charAt(index)、setCharAt(index, ch)、insert(index, ch)、reverse()、delete(start, end)、replace(start, end, String)

（22）StringTokenizer 类：构造(Str, delim),delim 默认为空格，用 hasMoreTokens( )和 nextToken( )遍历结果

（23）jar 文件：javap.exe 可将字节码反编译成源码，javadoc.exe 制作源文件类结构的 html 格式的文档（21）jar 文件会被移动到\jar\1.8.0_144\lib\ext\

（24）抽象类：有 abstract(可有)和非 abstract(可有)方法，有常量和变量。接口：只有 abstract 方法（但 JAVA8 之后可有 default 和 static 方法），只有常量。 选择区分在于是否希望有继承

（25）Scanner 类：用于正则化分割时的构造(String/File)，useDelimiter（"正则"）分割，用 hasNext()和 nextXxx()提取结果

（26）模式匹配：Pattern pat = P.compile(正则)， Matcher m = pat.matcher(Str), whlie(m.find()) -> println(m.group())。Matcher 对象其他函数：find(start)、matches( )判断是否匹配、replaceAll(Str)、replaceFirst( )、lookingAt( )看开头是否匹配

(27)第七章（除 Math 在 java.lang 中，BigInteger 和 BigDecimal 在 java.math 中其他都在 java.util 中）

【1】BigInteger(String doubleNum)、返回 BI，参数 BI->{ add、substract、multiply、devide、remainder、compareTo、.abs、.pow、toString(进制)可赋值 Str。 【2】格式化 NumberFormat。getInstance()， Str = nf.format(String / Decimal)

【3】HashSet<E>/TreeSet<E>都有 add(E)、contains(E)、size()、toArray( )、addAll(HS)并、retainAll(HS)交、removeAll(HS)差，遍历方法：toArray / Iterator<E> iter = HS.iterator( ) + iter.hasNext( ) + iter.next( ) 【4】 HashMap<K,V>/TreeSet<K,V>: get(key)、put(key, val)、remove(key)、containsKey(K)、containsVal(V)、clear()、clone()、size()、isEmpty()、values()返回 Vals 的 Collection、keySet()返回键的 Set 【5】Comparable 接口：类 E 要实现 public int compareTo(obj) {this - other \\ 升序} -> Arrays.sort(E[]) Comparator 接口：另定义实现 CPT 的 SortClass 内实现 public int compare(obj1, obj2) -> Arrays.sort(obj[], new SortClass())

（28）第八章 -> 线程:操作系统运算调度的最小的单位(新建，运行，中断，死亡-->优先级 1-10 递增默认为 5)。有 setName(Str)、Thread.currentThread.getname()、sleep()、interrupt()、wait()、notifyAll()、notify()、isAlive()。线程联合：在 threadA.run()里 start threadB 并 threadB.join()：先执行完 B 再继续执行 A。守护线程：在 thread.start()之前调用 setDaemon(true),无其他线程存活时就会死亡。 （29）sleep(): Thread 的静态方法，自动苏醒，可在任意位置写；wait(): Object 的实例方法、需要 notify 手动唤醒、只能写在 synchronized 块内。 （30）多线程：进程内的多线程工作，提高 cpu 利用率；线程调度：操作系统决定线程的时间片分配过程 （31）线程池 ExecutorService executor = Executors.newFixedThreadPool(int num); executor.execute(new RunnableClass()); executor.shutdown()。

第九章 -> 输入与输出全位于 java.io 包中（32）File 对象:构造(name) / (directory, name)。length()获取字节长度、exits()、getName()、getAbsolutePath、getParent()、isFile()、isDirecory()、delete()、mkdir()、createNewFile()。是目录对象时 list()和 listFiles()能够返回 String[]和 File[]的目录内文件数组。 文件过滤：class CS implements FilenameFilter { String Str; CS(String s) Str += s; public boolean accept(File dir, String name) return name.endsWih(Str); } CS cs = new CS(例："java"); file.listFile(cs)。筛 java 文

件 （33）打开可执行文件或执行命令：Runtime rt = Runtime.getRuntime(); rt.exec(file.getAbsolutePath() / 命令行); ++++++字符流 下述类有 read()、read(char[])、read(char[], offset, len)与对应的 write 方法+++++ （34）记得+++ try-catch +++【1】FileReader(String / File) 和 FileWriter(String / File), 有 read()和 write()单字符（返回 Unicode 值）或字符数组（返回读取数量）读写重载。 使用示例：file 和 str； char[] b = str.toCharArray(); FW output = new FW(file); output.write(b / str)； output.close()； FR input = new FR(file)； while((n = input.read(b, 0, 2)) != -1) res = new String(b, 0, n)； input.close()；【2】BufferReader(Reader)和 BufferWriter(Writer) 使用示例：BR input；BW output；while((str = input.readLine()) != null) { ouput.write(str)； ouput.newLine() } output.flush()、output.close()、intput.close、fr.close()、fw.close()。 ++++++字节流 下述类有 read()、read(byte[])、read(byte[], offset, len)与对应的 write 方法+++++ 【3】FileInputStream(String / File)和 FileOutputStream(String / File) 使用示例：byte[] b = str.getBytes()；FOS fos = new FOS(file)； fos.write(b)； fos.close()； FIS fis = new FIS(File)； while((n = fis.read(b, 0, 3)) != -1) str = new String(b, 0, n)。 【4】数据流： DataInputStream(FileInputStream)； DataOutputStream(FileOutputStream) :有 readInt()、readBoolean()等和 writeChar(char)等 使用示例： DIS dis = new DIS(fis)； while((ch = dis.readChar()) != “\0”) println(ch) 【5】Serializable 序列化接口，其中方法不可见故不需要实现，其方法会在序列化对象写入对象流时被 JVM 实现 【6】对象流（可用于深度克隆）： ObjectInputStream(InputStream)...使用示例：Class E implement Serializable { }; E a；FOS fos = new FOS(file)；Objos objos = new Objos(fos)； objos.writeObject(a)； FIS fis = new FIS(file)； Objis objis = new Objis(fis)； E b = (E)Objis.readObject()； close() 获得一个深拷贝对象。【7】上述例子中的 fos，fis 可以换成 ByteArrayOutputStream = new BAOS( )和 ByteArrayInputStream = new BAIS(fos.toByteArray( ))对象（记得 close），就不需要 file 对象。其本身有 write、read 单字节方法，配合 byte[ ]和 String.getBytes[] 【8】数组字符流 CharArrayReader -> read( ) 【9】随机读写流：RamdomAccessFile(String / File，”r” / ”rw”)；使用示例一（一个一个读）： RAF raf = new RAF(String, “rw”); for(int i : data) raf.writeInt(i); for(long i = 0; i< data.length; i++){ raf.seek(4*i); println(raf.readInt()) } raf.close(); 使用示例二（一行一行读）： raf.writeBytes(Str + “\n”); long len = raf.length(), pos = 0; raf.seek(pos); while(pos < len) { Str = raf.readLine(); pos = raf.getFilePointer(); println(Str) } 【10】文件锁 FileLock（RAF 必须是“rw”） 使用示例： RAF raf = new RAF(... ,”rw”); FileChannel fc = raf.getChannel(); FileLock lock = fc.tryLock(); 结束时 close()， lock.release()。 【11】字符串流 StringReader(String) / StringWriter( ), while((data = read()) != -1) (char)data; write(int / char[ ] / String)

（35）网络编程【1】URL(String 网址) / (String*3 协议、地址、资源)；使用示例： byte[] b; InputStream in = url.openStream(); while(n = in.read(b) != -1) Str = new String(b, 0, n); 【2】InetAddress 类 用 getByName(String) 构造 【3】网络套接字：端口号和 IP 地址组合；套接字连接: 客户端和服务端套接字对象用输入输出流链接。 TCP:使用示例： 服务器中 try{ ExecutorService pool = Executor.newFixedThreadPool(100); ServerSocket center = new SS(port); while(true) { Socket client = center.accept(); pool.execute(new Runnable(){Socket socket = client}) } } catch(){} finally { pool.shutdown() } 然后 Runnable 类中{ BufferReader -> InputStreamReader(socket.ggetInputStream, “UTF-8”); PrintWriter out = new PW(new OutputStreamWriter(socket.getOutputStream(), “UTF-8”))； while((line = BR.readLine()) != -1); out.println() } 客户端中 Socket socket = newSocket(“SERVER”, “PORT”); new Thread(()->{ try{ while((Str = BW.readLine()) != null) }catch(){} }).start(); 【4】UDP 使用示例： 服务器中： DatagramSocket server = new Dgs(port); while(true){ byte[] buffer = new byte[1024]; DatagramPacket dp = new DP(buffer, buffer.length); server.receive(dp); InetSocketAddress client = (InetSocketAddress) dp.getSocketAddress(); pool.execute(new ClientHandler(dp, server,client)) } class ClientHandlerimplements Runnable { 接收 packet,server, client, run(){ String = new String(packet.getData(), 0, packet.getLength()) 发送： byte[] resp = str.getBytes(StandardCharsets.UTF-8) ; DatagramPacket pac = new DP(resp, resp.length(), client.getAddress(), client.getPort()); server.send(pac) } } 客户端中： 接收： DatagramSocket socket = new DatagramSocket(); InetAddress addr = InetAddress.getByName("127.0.0.1"); new Thread(()->{ while(true) { DatagramPacket resp = new DatagramPacket(buf, buf.length); socket.receive(resp); String reply = new String(resp.getData(), 0, resp.getLength(), StandardCharsets.UTF_8);} }) 发送： byte[] data = Str.getBytes(); DatagramPacket packet = new DatagramPacket(data, data.length, addr, 8888); socket.send(packet); 最后： socket.close()