|  |  |
| --- | --- |
| **①java**摒弃指针与内存管理,使用字节码在虚拟机运行,完全面向对象,支持多线程,动态分配内存,公用静态变量代替全局变量,不用goto,不支持头文件与宏定义,数据类型分配固定长度**javac.exe** 编译器 java.exe 解释器 jdb.exe 调试器 javap.exe反编译**javaSE**:标准版,桌面和服务器应用程序 **javaEE**:企业版 **javaME**:小型版,嵌入式 **命令行编译：**javac Hello.java →生成 Hello.class→运行 java Hello **②标识符**第一字符不能是数字,长度不受限制,区分大小写 **long**数值后接L,float类型后接f,小数位保存6位 **%m.nf**,float占m列，保存小数n位 **③精度**就高不就低,就长不就短 **=**左边必须是变量 **a<<n**时,byte或short强制转换为int,%32后高位用符号位补充,long则%64 **a>>n**时,移进高位用符号位填充 **^**按位异或 **x=1**时,((y=1)==0)&&((x=6)==6) x=1, ((y=1)==0)&((x=6)==6) x=6 **④封装**:数据属性与操作功能封装 **继承:**继承父类属性与功能 **多态**:操作名称相同接收消息不同,或同一操作不同类型对象调用行为不同 **类**是java程序基本要素,封装了object的变量与方法,类名符合标识符要求 **内存空间**存放对象实体,对象中存放引用,引用相同则实体相同 **类(成员)变量**也叫静态变量,其保存在全局数据,在该类被加载到内存时，就分配了相应的入口地址区**final**修饰的常量不占用内存,必须初始化 **方法名字**首字母小写,后面单词首字母大写 **类中方法**必须有方法体 **方法**所有参数都是传值的,对象,数组,接口为传引用 **重载**只关心参数个数及参数类型 **静态方法**中不可用this **不能**用protected修饰类,不能用private修饰外部类,只能修饰内部类 **类是对象**的一个集合,对象是类的实例 **类是**一个抽象的概念,对象是类的一个具体,类是一个静态的概念,对象是一个动态的概念 **包名**全部由小写字母(多个单词也全部小写),一般由倒置的域名开头 **⑤继承**有利于代码的重用 **friendly**也就是默认权限 **子类构造方法**总是先调用父类的(不继承父类构) **当访问数组**中不存在的元素时,会产生数组下标越界 **运行时异常:**RuntimeException空对象异常(NullPointerException),算术异常(ArithmeticException),类型强制转换异常(ClassCastException),数组下标越界异常(ArrayIndexOutOfBoundsException),数值格式异常(NumberFormatException) **父子类**方法相同类型不同可形成重载 super调用父类构造方法或者隐藏的成员变量,方法 **final**类不能被继承 **父=子**:上转型,子=父:下转型(方法参数也可上转型) **上转型**丢失了一些属性与功能,将其转换为子类又拥有了 **abstract**类构造方法为protected,方法为abstract不能实现,不能用new创建对象 **java**不支持多重继承 **interface**定义接口,implements声明,常量用public static final(可省略),方法用public abstract(可省略) **实现接口方法**用public修饰(接口回调使得可以产生不同行为),不可省略 **extends**使得接口可被继承 **内部类**无静态变量/方法(static内部类除外) **泛型类/接口**只能调用object类中方法 **⑥String**(char a[])用字符数组创建string对,**String**(char a[], int startIndex, int count)截取字符,int **compareTo**(String s):字典序比较大小,=s0,>s正<s负,int **indexOf**(String s):检索s出现位置,**substring**(int startpoint):获取当前串:startpoint:位置之后子串,**replaceAll**(String s1, String s2):s2替换s1,**trim**():去掉字符串中空格:**parseInt**/Byte/Short/Long/Float/Double(string):数据相互转化, String **valueOf**(byte/int/long/float/double b),转化**tostring**方法实现:**public String** toString() return name+": "+score; void **getChars**(int start, int end, char c[], int offset):部分字符串转字符数组char[] **toCharArray**():转字符数组**String**(byte[]) **String**(byte[], int offset, int length) byte[] **getBytes**() ”Hello world”.substring(6)返回:world;substring(6,9)返回wor。Integer.parseInteger(“123”)。String.valueOf(1234);Integer.toString(123) **StringBuilder**线程不安全，StringBuffer线程安全。StringBuilder比StringBuffer效率高**StringBuffer**():无size时为16,参数为string时size=string的长度+16 **capacity**():获取stringbuffer实际容量 **append**(string) **charAt**(index) **setCharAt**(index,char ch) **insert**(index, str):返回插入后引用 **reverse()**:返回翻转后引用 **delete**(startindex,endindex) **replace**(startindex,endindex,str) **StringTokenizer**(str,String delim):分隔符为delim的分析器,利用hasMoreTokens()来判断循环,nextToken()取结果 **Scanner**(system.in):输入 scanner.useDelimiter(“正则”):scanner分隔符标记,后续利用while(scanner.hasNext()),scanner.nextDouble() **Pattern** p=Pattern.compile(“正则”),Matcher m=p.matcher(“字符串”),while(m.find()),str=m.group() **相同参数类型**的泛型类和派生类可相互转换 **\\d**表0-9,\\D表非数,\\s表空格类字符,\\S非空格类字符,\\w标识符字符(/$),\\W不能用于标识符字符,?0或1次,{n}恰好出现n次,{n,}至少,{n,m}n到m次,[abc]abc中任意一个字符,[^]除之外字符,[a-b]a到b任意一个,p{Alpha}表字母字符 **str.replaceAll**(正则,”替换”) **string[] str.split**(正则)利用正则分割 **⑦线程**是比进程更小的执行单位,一个进程可以有多个线程 **多线程**就是在单个程序中同时运行多个线程来完成不同的工作 **线程间**实例的域变量是共享的 **Date**无参构造获取当地时间 long **currentTimeMillis**()获取系统走过毫秒数SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss(a)(EE) "); sdf1.format(Date对象) **Calendar** calendar = Calendar.getInstance();set(year, month, date, hour, minute, second) calendar.get(Calendar.MONTH)获取月份/.. **Math**类abs(a) max(a,b) min(..) random()范围[0,1) pow(a,b)ab sqrt(a) log(a) sin(a) asin(a)反正弦**BigInteger**(string val) add(Biginterger) substract(Big) multiply(Big) divide(Big) remainder(Big) CompareTo(Big) abs() pow(exp) tostring(int)p进制字符串表示NumberFormat f = NumberFormat.getInstance() ; setMaximumFractionDigits(int newValue)最大小数位数/setMinimumFractionDigits /setMaximumIntegerDigits最大整数位数/ setMinimumIntegerDigits f.format(int) **LinkedList<String>** mylist = new LinkedList<String>();链表mylist.add("How")插表尾; add(index,element) addfirst(element) removeFirst,removeLast get(index)/getFirst/getLast indexOf(element)返回首次出现位置 lastIndexOf(element)最后出现位置 set(index,element)替换位置节点 contains(element)判断是否含有 clone()克隆**Iterator**<Student> iter = mylist.iterator();while(iter.hasNext())Student temp = iter.next(); **HashSet**<String> set = HashSet<String>(); 集合不允许有相同元素 set.add(b)相同则无效 clear(),contains(object),isEmpty(),remove(object),size(),Object[] toArray()集合元素存放到数组中 containsAll(hashset set)判断当前集合是否含有参数集合 clone() addAll()并,retainAll交,removeAll()差 **HashMap<k,v>** k不能发生逻辑冲突 clear(),clone(),containsKey(key) containsValue(value),get(key)返回键,isEmpty(),remove(key),size(),Collection<Integer> collection = map.values(); Iterator<Integer> iter = collection.iterator();遍历 **TreeSet**<E>树集 add(o),clear(),contains(o),first()返回第一个(最小)对象,last()返回最后(最大)对象 isEmpty(),remove(o),size() class Student implements **Comparable**{ public int compareTo(Object o) Student stu = (Student)o; if(this.score==stu.score) return 1; return (this.score - stu.score); class Student**Comparator**(不同于student类) implements Comparator{ public int compare(Object o1, Object o2) { return ( ((Student)o1).score - ((Student)o1).score); }} main函数中Arrays.sort(students, new StudentComparator()); comparator算法与数据分离**TreeMap**<K,V>(Comparator<K> comp或无)其常用方法与hashMap类似 **Stack**<E> peek()获取顶端数据,search(o)获取堆栈位置 **⑧InputStreamBuffer**不属于字节流 **4个生命周期**:新建,运行(ILLegalThreadStateException),中断(sleep,wait,中断),死亡**setPriority**(int grade)设置优先级(lllegalArgumentException(>10)) **高优先级**的线程能始终运行class Bank implements **Runnable**{ public void run() { Thread.currentThread().getName().equals() }} **不同线程**的run()方法中的局部变量互不干扰**isAlive**(),start后死亡前都为true **一个已经运行**的线程在没有进入死亡状态前，不要再给线程分配实体**currentThread()**,sleep()是静态方法interrupt()吵醒休眠线程(InterruptedException异常结束休眠) public **synchronized** void saveOrTake(int number){ **wait(),notify(),notifyAll**()是final方法**hangUP();**挂起target.restart();重启join()**线程联合** **ajion了b**会等待b运行完(InterruptedException) thread.setDaemon(true)将自己设置成守护线程,跟随别人一起强制结束自己 **⑨所有事件类**的父类是AwtEvent InputStream, OutputStream (**字节输入/出流**) Reader, Writer(**字符输入/出流**) File(String filename); File(String directoryPath, String filename); getName()获取文件名字,canRead(),canWrite(),exits(),length()字节长度,getAbsolutePath(),getParent(),isFile(),isDirectory(),isHidden(),lastModified() mkdir()创建目录String[] list(FilenameFilter obj)字符串数组返回目录下指定类型的全部文件/无参返回全部 | |
| class MyException **extends Exception** String message;**MyException(int n)**{message = n + ": not a positive number";public String getMessage() return message;  class A  public void f(int n) **throws MyException**  if(n<0)MyException ex = new MyException(n);throw(ex); // 抛出异常，结束方法f的执行  **try**{ Thread.sleep(800); }catch(InterruptedException e) {}  **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。不过该过程要抛出异常，  还要调用 while(scanner.hasNext()){}  如果是 scanner.nextInt(); 还有 InputMismatchException 异常  不想是空格也可以正则表达式 scanner.useDelimiter(正则表达式); 。  模板：try{  FileReader fr = new FileReader("input.txt"); BufferedReader input = new BufferedReader(fr); FileWriter fw = new FileWriter("output.txt"); BufferedWriter output = new BufferedWriter(fw); String 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(); 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 称为 Swing 组件，其中大部分组件是轻组件 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 }  class MyWindow extends JFrame  {JTextField text; PoliceStation police;  MyWindow(){setLayout(new FlowLayout());  police = new PoliceStation();  text = new JTextField(10); add(text); text.addActionListener(police); //text是事件源，police是监视器setBounds(100,100,150,150);setVisible(true);validate();setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);}  class PoliceStation implements ActionListener  public void actionPerformed(ActionEvent e)  String str = e.getActionCommand();  System.out.println(str);  System.out.println(str.length());  public void actionPerformed(ActionEvent e)  JTextField textSource = (JTextField)e.getSource();  if(textSource==titleText)  this.setTitle(titleText.getText()); | 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)  ServerSocket server = new ServerSocket(4333) IOExceptio  Socket socketAtServer = null;  socketAtServer = server.accept(); //等待建立连接  in=newDataInputStream (socketAtServer.getInputStream());  out=newDataOutputStream(socketAtServer.getOutputStream());  out.writeUTF(“”) in.readUTF(“ ”);  客户端 Socket socketAtClient; = new Socket("localhost", 4333)  DataInputStream in = new DataInputStream (s~t.getInputStream());  DataOutputStream out=newDat~am(s~t.getOutputStream());  str = in.readUTF();  UDP 数据报  接收：DatagramPacket pack = null; //UDP 数据报  DatagramSocket mail = null;  byte b[] = new byte[8192];  try {  pack = new DatagramPacket(b,b.length);  mail = new DatagramSocket(3456); //尝试建立连接}  catch(Exception ex) {}  while(true) {  try {  mail.receive(pack); //接收其他 pack  String message = new String(pack.getData(),0,pack.getLength());  inMessage.append("\nClientB:\n"+message);  inMessage.setCaretPosition(inMessage.getText().length());}  catch(Exception ex) {}}  发送 byte b[] = outMessage.getText().trim().getBytes();  try {  InetAddress address = InetAddress.getByName("127.0.0.1");  DatagramPacket  date = new DatagramPacket(b,b.length,address,1234); //发的地址  DatagramSocket mail = new DatagramSocket();  mail.send(date); //发送信息 |