|  |  |
| --- | --- |
| **树**中所有节点的度等于所有节点数-1 **在一棵**完全二叉树中,假定根节点编号为0,则对于编号为i的节点,其双亲节点的编号为i-1/2 在一棵数的静态双亲表示中，**每个**存储节点包含2个域 **向具有**n个节点的堆中插入一个新元素的时间复杂度为O(log­2n) **根节点**的度为2 **在一棵**高度为h的理想平衡树中,最少有2h个节点,最多有2h+1-1个节点 | |
| 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 |