|  |  |
| --- | --- |
| **每**一种支持 Java 应用程序的硬件平台和操作系统都拥有它自己的Java 虚拟机。**JRE**包括JVM、java核心类库以及一些支持文件。**标识符**以字母\_$开头，大小写敏感。**两个**相加默认为int，赋值给byte和short时要显示转换。**long**后面加L。**float**后必须加f，缺省默认为double。**科学计数**123e13。flaot32位double64位。**bool**4字节。**char**16位unicode char c=’\u????’4个16进制数。int i=’a’或者float正确。**自动转换**优先级由低到高，可自动从低转换到高byte**short**char**int**long**float**double。自动转换规则表 数据类型(char)(byte**short**)(byte**short**int)(byte**short**int**long**)(byte**short**int**long**float)可以自动转换的数据类型(int)(int)(long)(float)(double)。**高到低**则需要显示转换。**数组声明**x[] a;x a[];x[][] a;x a[][]。**i=3**;j=4;j=i++ +j+ ++i后i=5j=12。**逻辑**运算符&&为短路&不为短路。**优先级**()自左至右结合[]自右至左**从高到低**(.()[])[正号负号++--~!](\*/%)(+-)(<< >> >>>)(< <= > >=)(== !=)(&)(|)(^)(&&)(||)[:?][= += ……]**switch**表达式中只能是byteshortintchar类型的。**内部类**实例化Outer outer = new Outer();Outer.Inner inner = new outer.new Inner();**interface**中变量默认为public final static,方法默认为public abstract。**chatAt(i)**返回第i个字符，从0开始。**string1.getChars(0,len,tempArray,0)**将String转化为char数组。**string1.concat(string2)**连接。忽略大小写比较**equalsIgnoreCase()**。**”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。**ArrayList**线程不安全,**Vector**线程安全。**LinkedList**线程不安全，同步列表List l = new synchronizedList(new LinkedList());**Set**线程不安全，同步类似lisk.。**HashSet<E>: add**添加**.addAll(set)**并运算**.retainAll(set)**交运算**.removeAll(set)**差集。**TreeSet<E>** 其中的元素要实现Comparable接口，实现方法public int compareTo(…)。TreeSet和HashSet没有get()方法，只能使用迭代器访问元素，TreeSet<Person> ts=new TreeSet<>();......Iterator<Person> it=ts.iterator();while(it.hasNext()){sout(it.next);}HashMap<K,V>线程不同步。访问方式HashMap<String,Person> m=new ;Collection<Person> collection=m.values();Iterator<Person> it=collection.iterator();while(it.hasNext()){}。**异常**指在运行时代码序列中产生一种异常情况，换句话说，异常是一个运行时错误。**运行时异常**都是RuntimeException 类及其子类异常，如空对象异常(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)等。这些异常是不检查异常，程序可以选择捕获处理，也可以不处理。这些异常一般是由程序逻辑错误引起的，程序应该从逻辑角度尽可能避免这类异常的发生。**非运行时异常**是RuntimeException以外的异常，类型上都属于Exception 类及其子类，如 IOException、SQLException 等。这些异常是检查异常，从程序语法角度讲是必须进行处理的异常，如果不处理，程序就不能通过编译。一般情况下不自定义检查异常。**字节流InputStream**，int read()返回读取的一个字节，到达末尾返回-1。int read(byte[] b)。int read(byte[] b,int off,int len)。close()。**OutputStream**，void write(int b)，写入低八位。void write(byte[] b)。void write(byte[] b,int off,int len)。**FileInputStream**，public FileInputStream(String name) throws FileNotFoundException，public FileInputStream(File file) throws FileNotFoundException，**FileOutputStream**多一个参数Boolean append**管道流**，PipedInputStream,PipedOutputStream，在使用管道前，两者必须进行连接。(1)PipedInputStream(PipedOutputStream pos);PipedOutputStream(PipedInputStream pis); (2)在类 PipedInputStream 中，connect(PipedOutputStream pos);在类PipedOutputStream中，connect(PipedInputStream pis);**数据流**DataInputStream(InputStream in)，DataOutputStream(OutputStream out)。**字符流**，Reader类，int read()读取单个字符，int read(char[] buf),int read(char[] cbuf,int off,int len)。Reader类，void write(int c),void wirte(char[] buf),void write(char[],int off,int len),void write(String s)。**FileReader**, public FileReader (String name) throws FileNotFoundException，public FileReader (File file) throws FileNotFoundException。**FileWriter**多一个参数Boolean append。**缓冲流**,**BufferedReader(FileReader f)** (str=br.readline())!=NULL,bw.newLine()。**线程设置优先级**setPriority()默认为5，级别1-10。**本机地址**，InetAddress l=InetAddress.getLocalHost();=InetAddress.getByName(“www.”);**UDP**,DatagramSocket(int port),DatagramPacket(byte[] data,int length,InetAddress r,int remotePort)。**TCP**,Socket s=new Socket(“localhost”,port),ServerSocket ser=new (port),s=ser.accept(),**半关闭**,s.shutdownOutput(),**URL**,URL u=new URL(“http://”),  FileReader:FileNotFoundException。File file=new File(“D:\a.txt”);file.createnewFile():IOException  wait() sleep() join():InterruptedException  队列java.util.Queue Queue<String> q=new LinkedList<String>();q.offer(“a”);添加q.poll()返回第一个元素，并在队列中删除q.peek()返回第一个元素 | |
| import java.util.HashMap;import java.util.Map;import java.util.Objects;  public class SparseMatrix {  private static class SparseEle {  @Override  public boolean equals(Object o) {  if (this == o) return true;  if (o == null || getClass() != o.getClass()) return false;  SparseEle sparseEle = (SparseEle) o;  return rowIndex==sparseEle.rowIndex&&colIndex==sparseEle.colIndex;}  @Override  public int hashCode() {return Objects.hash(rowIndex, colIndex);}  import javax.swing.\*;import java.awt.\*;import java.awt.event.ActionEvent;import java.awt.event.ActionListener;import java.util.HashMap;import java.util.Map;  public class Translation implements ActionListener {  static Map<String,String> dict = new HashMap<>();  JButton button;  JTextField textField;  JLabel label;  public Translation() {dict.put("hello","你好");}  public Container createPane() {  button = new JButton("中<->英");  textField = new JTextField();  label = new JLabel();  Container pane = new Container();  Container buttonPane = new Container();  pane.setLayout(new GridLayout(1,3));  buttonPane.setLayout(new BorderLayout());  pane.add(textField);  buttonPane.add(button,BorderLayout.CENTER);  pane.add(buttonPane);  pane.add(label);  button.addActionListener(this);  return pane;}  public static void ShowGUI() {  Translation translation = new Translation();  JFrame frame = new JFrame("Translation");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setSize(350,90);  frame.setLocation(200,50);  frame.setContentPane(translation.createPane());  frame.setVisible(true);}  @Override  public void actionPerformed(ActionEvent actionEvent) {  String text=textField.getText().toLowerCase();  if(text.equals("")) {  return;  }  boolean flag=false;  for(Map.Entry<String,String> entry:dict.entrySet()) {  if(text.equals(entry.getKey())) {  label.setText(entry.getValue());  flag=true;}  else if(text.equals(entry.getValue())) {  label.setText(entry.getKey());  flag=true;}}  if(!flag) {  if(text.matches("\\w+")) {  label.setText("抱歉，我不知道");}  else {  label.setText("Sorry, I don't know.");}}}  public static void main(String[] args) {  javax.swing.SwingUtilities.invokeLater(new Runnable() {  @Override  public void run() {  ShowGUI();}});}}  public void getContent() throws Exception{  URL web = new URL("http://search.cnki.com.cn/Search/Result?content=%u8BA1%u7B97%u673A");  InputStream inputStream = web.openStream();  byte[] bytes = inputStream.readAllBytes();  content = new String(bytes);  inputStream.close();}  import java.io.IOException;import java.net.\*;import java.util.HashMap;import java.util.Map;  class ServerTest {  public static void main(String[] args) throws Exception{  UDPServer server = new UDPServer();  while (true) {  server.receiveUDP();  Thread.sleep(500);}}}  public class UDPServer {  HashMap<String,Integer> nameANDport = new HashMap<>();  DatagramSocket server;  public UDPServer() {  try {server = new DatagramSocket(1234);}  catch (SocketException e) {e.printStackTrace();}}  public void addUser(String name,int port) {  nameANDport.put(name,port);}  public void receiveUDP() {  byte[] data = new byte[1024];  DatagramPacket datagramPacket = new DatagramPacket(data,data.length);  while (true) {  try {  server.receive(datagramPacket);  } catch (IOException e) {  e.printStackTrace();  }  String content = new String(data,0,datagramPacket.getLength());  String[] info = content.split("/");  switch (info[0]) {  case "add":  addUser(info[1],datagramPacket.getPort());  break;  case "send":  sendUDP(info[1],info[2]);}}}  public void sendUDP(String senderName,String message) {  DatagramPacket messagePacket = null;  for(Map.Entry<String,Integer> entry: nameANDport.entrySet()) {  String name = entry.getKey();  int port = entry.getValue();  if(!senderName.equals(name)) {  String m = senderName+":"+message;  byte[] temp = m.getBytes();  try {messagePacket = new DatagramPacket(temp,0,temp.length,InetAddress.getLocalHost(),port);}  catch (UnknownHostException e) {e.printStackTrace();}  try {server.send(messagePacket);}  catch (IOException e) {e.printStackTrace();}}}}} | Map <String,String>map = new HashMap<String,String>();  map.put("熊大", "棕色");  map.put("熊二", "黄色");  for(Map.Entry<String, String> entry : map.entrySet()){  String mapValue = entry.getValue();  String mapKey = entry.getKey();  System.out.println(mapKey+":"+mapValue);} |