第一章：java 语言概述
诞生于 1995，一种可移植的、跨平台的语言
特点：面向对象，可移植，健壮，安全，动态
Java SE 桌面开发和低端商务应用的解决方案。
Java EE 是以企业为环境而开发应用程序的解决方案。
Java ME 是致力于消费产品和嵌入式设备的最佳解决方案。
JVM Java 虚拟机 JDK 开发工具包 JRE Java 运行环境 IDE 集成开发环境
.java 文件是源文件，通过 javac 命令编译后生成.class 文件；.class 文件是字节码结文件，即.java 文件编译后的代码。
第二章：数据类型，运算符，表达式和语句
标识符：字母是区分大小写的，不能是 true, false, null（尽管它们不是关键字）
关键字：被赋予特定意义，abstract, continue, for, new, switch, default, goto*…
8 种基本数据类型可分为 4 大类型：　　逻辑类型：boolean
字符类型 char 整数类型：byte(1), short(2), int(4), long 浮点类型：float(7), double
字符类型: Uincode"\uxxx' 2 个字节，16 位，最高位不是符号位 范围：0~2^16-1
（char 的最高位不是符号位，有可能超出 short 的取值范围，转化用 int）
定义数组 int intArray[ ][ ] = { {1,2}, {2,3}, {4,5} };　int a[ ][ ] = new int[2][3];
Instanceof 左对象，右类 boolean f = rectangleOne instanceof Rect;
第三章：类与对象
面向对象编程主要有三个特性：封装，继承，多态
成员变量的类型，对象接口也可以，　类内有效与书写的先后位置无关
构造方法：不能被 static、final、synchronized、abstract 和 native 修饰
Java 具有"垃圾收集"机制，Java 的运行环境周期地检测某个实体是否已不再被任何对象所引用，如果发现，就释放该实体占有的内存。
Static：静态变量既可以通过某个对象访问也可以直接通过类名访问，不可 this
Final：修饰的成员变量不占用内存，必须要初始化。有 static final a;
静态方法(类方法)只能调用该类的静态方法用静态变量 main 方法也是静态方法
在 Java 中，方法的所有参数都是"传值"的,方法中参数变量的值是调用者指定的值的拷贝。方法如果改变参数的值，不会影响向参数"传值"的变量的值.而当传的是引用类型数据 包括对象、数组、接口（interface）。当参数是引用类型时，"传值"传递的是变量的引用。
使用 import，增加编译时间，但不会影响程序运行的性能。
访问权限：在与类 A 同 package 的另外一个类 B 中，可以访问对象 a 的以下成员　Friendly (or default), protected, public
在类 A 的子类 B 中(不同 package)，可以访问对象 a 的以下成员
protected, public
第四章：继承（extends）
子类与父类：不支持多重继承，继承的变量和方法按照访问权限,例如不在同一个包里，子类不能继承父类的 friendly
子类不继承父类的构造方法，可采用 super()来表示
final 类不能被继承，final 方法 不能被重写
abstract 类不能用 new 运算符创建对象，必须产生其子类，由子类创建对象。
如果 abstract 类的类体中有 abstract 方法，只允许声明，而不允许实现；而该类的非 abstract 子类必须实现 abstract 方法，即重写 (override) 父类的 abstract 方法。
接口 interface+接口的名字不许提供方法的实现，接口也可以被继承 (extend)
方法 public abstract int g(int x,int y);
接口中的常量用 public static final 来修饰，但可以省略
接口中的方法用 public abstract 来修饰，但可以省略
在实现接口中的方法时，一定要用 public 来修饰，不可以省略
接口回调：接口回调是多态的另一种体现。把子类创建的对象的引用放到一个父类的对象中时，得到该对象的一个上转型对象
内部类：类体中不可以声明静态变量（类变量）和静态方法（类方法）
匿名类：一定是内部类，主要用途就是向方法的参数传值
Java 泛型的主要目的是可以建立具有类型安全的数据结构，如链表（LinkedList）、散列映射（HashMap）等数据结构。
第五章 字符串
字符串类 String 表示一个 UTF-16 格式的字符串
public int length()　获取一个字符串的长度
public static int parseInt | Byte Short Long Float Double(String s)
public String valueOf(byte b)
public String toString(){} 返回该类输出的字符串
public byte[] getBytes(): 将当前字符串转化为一个字节数组
StringBuffer 类：能创建可修改的字符串序列
Buffer append 方法：可以将其它 Java 类型数据转化为字符串后
char charAt(int index)　void setCharAt(int index, char ch)
StringBuffer（多线程）是线程安全的 StringBuilder（效率高）不是线程安全的
线程安全是指多个线程操作同一个对象不会出现问题
StringTokenizer fenxi = new StringTokenizer(s,".");
for(int i=0; fenxi.hasMoreTokens(); i++)　{String str = fenxi.nextToken();}
String cost = "市话费: 176.89 元，长途费: 187.98 元，网络费: 928.66 元";
Scanner scanner = new Scanner(cost);
scanner.useDelimiter("[^0123456789.]+");
　while(scanner.hasNext()){
　try{　double price = scanner.nextDouble();
　System.out.println(price);}
　catch(InputMismatchException exp)
　{　String t = scanner.next(); }}
.任意字符 \\d 0-9 \\D非数字 \\s 空格类\\S 非~\\w可标识符\\W
X? 0|1 X{n}恰好出现 n 次 X{n,}至少 n 次　X{n,m} n-m 次
"[159]ABC" "1ABC" 、"5ABC"和"9ABC"都是行
[^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
Pattern p;　Matcher m;
p = Pattern.compile("\\d+"); m = p.matcher("2008 年 08 月 08 日");
while(m.find()) { String str = m.group();}
第六章，泛型和集合
回一个整数，0 表示一月，1 表示二月 calendar.get(Calendar.MONTH);
求相差日期 calendar.set(1931,8,18); long timeOne = calendar.getTimeInMillis();
LinkedList<E>：add() indexof()　<E>,get(index)
HashSet<E>　add() clear()　remove()　contain()
Iterator<Student> iter = tempSet.iterator();　while(iter.hasNext()){}
HashMap<K,V>: get(Object key)
class StudentComparator implements Comparator
{
　int score;
　public int compare(Object o1, Object o2){
　　return (((Student)o1).score - ((Student)o1).score);}}
　main(){Arrays.sort(students, new StudentComparator());}
class A{
　public int compareTo(Object o) {return (this.score - stu.score);}
Stack<E> push() pop() intValue()
第七章 线程
线程是比进程更小的执行单位。一个进程在其执行过程中，可以产生多个线程，形成多条执行线索，动态概念
JVM 在主线程和其他线程之间轮流切换，保证每个线程都有机会使用 CPU 资源
四种状态：　新建　运行　中断　死亡
Thread 的子类创建线程 写一个类继承 Thread 重写 public void run 方法 主线程 new 一个类，然后 start 就可以
实现 Runnable 接口的类的实例，然后直接 Thread(Runnable target) start 就可
if(Thread.currentThread().getName().equals(name1)){}是里面的一个好语句
try{ Thread.sleep(1000); }catch(InterruptedException e) {}使得都能调用
一个已经运行的线程在没有进入死亡状态时，不要再给线程分配实体。
如果实现 runnable 类里面有 Thread 变量，可以用 thread1.interrupt();唤醒
　线程同步是指多个线程要执行一个 synchronized 修饰的方法，如果一个线程 A 在占有 CPU 资源期间，使得 synchronized 方法被调用执行，那么在该 synchronized 方法返回之前（即 synchronized 方法调用执行完毕之前），其他占有 CPU 资源的线程一旦调用这个 synchronized 方法就会引起堵塞，堵塞的线程要一直等到堵塞的原因消除（ 即 synchronized 方法返回）
public synchronized void saveOrTake(int number){}
使用 wait()方法可以中断方法的执行，使本线程等待，暂时让出 CPU 的使用权，并允许其它线程使用这个同步方法。其它线程如果在使用这个同步方法时不需要等待，那么它使用完这个同步方法的同时，应当用 notifyAll()方法通知所有由于使用这个同步方法而处于等待的线程结束等待。
线程联合：B.join()如果线程 A 在占有 CPU 资源期间一旦联合 B 线程，那么 A 将立刻中断执行，一直等到它联合的线程 B 执行完毕，A 线程再重新排队等待
守护线程：当程序中的所有用户线程都已结束运行时，即使守护线程的 run()方法中还有需要执行的语句，守护线程也立刻结束运行
thread.setDaemon(true);

服务器端 ServerSocket server = new ServerSocket(4333) IOExceptio
　Socket socketAtServer = null;
socketAtServer = server.accept();　//等待建立连接
in = new DataInputStream(socketAtServer.getInputStream());
out= new DataOutputStream(socketAtServer.getOutputStream());
out.writeUTF("") in.readUTF(" ");
客户端 Socket socketAtClient;　= new Socket("localhost", 4333)
DataInputStream in=newDataInputStream(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
}

第八章 输入输出流
java.io 中有 4 个重要的 abstract class
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 read2 = new File("C:\\Users\\slark ","userTotTime.txt"); //文件路径
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);
如="[^0123456789.]+"解析出来就只剩下数字。
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();
第九章 图形用户界面设计
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 包为我们提供了更加丰富的、功能强大的组件，称为 Swing 组件，其中大部分组件是轻组件，没有同位体，而是把与显示组件有关的许多工作和处理组件事件的工作交给相应的 UI 代表来完成。
　这些 UI 代表是用 Java 语言编写的类，这些类被增加到 Java 的运行环境中，因此组件的外观不依赖平台，不仅在不同平台上的外观是相同的，而且与重组件相比有更高的性能。
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 }
第十章 URL 网络编程
URL 对象通常包含最基本的三部分信息：协议、地址、资源。
常用的 http、ftp、file 协议都是 Java 虚拟机支持的协议
地址必须是能连接的有效的 IP 地址或域名（host name）
资源可以是主机上的任何一个文件
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 Hyperlink**Listener 方法 void hyperlinkUpdate(HyperlinkEvent e)**
InetAddress 类的对象含有一个 Internet 主机地址的域名和 IP 地址，例如：www.sina.com.cn/202.108.35.210
套接字 Socket ： 端口号与 IP 地址的组合得出一个网络套接字。
端口号被规定为一个 16 位的整数 0~65535。其中，0~1023 被预先定义的服务通信占用（如 telnet 占用端口 23，http 占用端口 80 等）。除非我们需要访问这些特定的服务，否则，就应该使用 1024~65535 这些端口中的某一个进行通信，以免发生端口冲突。
所谓套接字连接就是客户端的套接字对象和服务器端的套接字对象通过输入、输出流连接在一起

·第一章:Java 运行平台：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。JRE（运行环境）JDK（开发工具包）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 →生成 Hello.class→运行 java Hello。**简述如何搭建Java 开发环境**:首先下载安装JDK 然后配置环境（1）配置PATH， 操作系统运行环境的路径（2）配置CLASSPATH JAVA 运行应用程序时所需要的类包的路径 （3）配置JAVA_HOME 供需要运行 JAVA 的程序使用.

·第二章:1.java 文件->编译->class 文件->java 虚拟机,解释机器码(java.exe)->程序 2.标识符:字母数字_$不能数字开头 3.关键字:enum、transient、strictfp、volatile、native、(goto,const 保留字)4.数据范围:077(八)0x3ABC（十六）5.Byte1 字节 8 位 -2^7~2^7-1 (byte 运算必须转化 byte s = byte（a+b）) 6.short2 字节 16 位-2^15~ 2^15-1 int4 字节 32 位-2^31~2^31-1 int32 位 -2^31~2^31-1 7.long8 字节 64 位-2^63~2^63-1 (后面加 l 或 L) boolean 单个 32 位、数组中 8 位 8.char2 字节 0~65535(0~2^16-1)('1'、转义字符、'\u????'(四位十六进制)使用时是一个数字)'t'水平制表符'n'换行'f'表格符'r'回车'\"'双引号'\'单引号'\\'反斜线 9.float4 字节 1.4E-45 ~ 3.4E+38(后面要加 f) 10.double8 字节 4.9E-324~1.7E308(后面加 d/D 可缺省)Long.Size 获得位数 11.默认值:数据的都是 0,char 是'\u0000',String 是 null,boolean 是 false 精度级别:byte short int long float double(左到右自动转换，右到左显式转换)(130 转化为 byte00000001000010->10000010(负数)->绝对值 01111110(126)->真实值-126)(float->int 不会四舍五入) 12.System.out.printf("%5.2f")输出小数部分最多 6 位的 float 数据，小数点后 2 位(会四舍) 常量值 1 到常量值 n 也必须是整型或字符型 14.数组声明不可，要使用数组要分配空间。数组下标访问 ArrayIndexOutOfBoundsException 15.%可以对浮点数用，3.5%2 = 1.5，++可以对浮点型用 17.Instanceof 确定某个对象是否属于某个类 18.<<左移右补 0>>右移左按符号补 0 或 1>>>不带符号右移左补 0 19.switch 中表达式是整型/字符型； 20.A.基本（简单）数据（注意 String 不是）：1.数值：整型 btye、short、int、long；浮点 float、double；2.字符（char）；3.布尔（boolean）B.复合数据：类接口数组

·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(运行出错) 4.final 修饰的常量,不占内存,不能通过类名访问。用 static 修饰的成员变量称为静态变量（类变量） 5.重载:参数列表不同即可.(返回值可以不同) 6. package cn.edu.szu.javapd.pwk; 13.java.lang.Number 中有基本类型的类包装 ByteShortLongInteger FloatDouble Character. 7.面向对象好处：模块化、信息隐藏、代码重用、易调试.8.Java 中只有按值传递，没有按引用传递 9.this 代表对当前对象的引用，不能出现 10.在类方法中；在构造方法中使用 this 调用其他构造方法时须放在方法第一 11.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

·第四章:继承与接口 1.重写/覆盖：名字返回类型参数个数和类型全相同。2.重载是编译时处理覆盖是运行时处理.3.访问权限要大于等于被覆盖方法的权限，例外列表要小于等于被覆盖方法的例外列表。9. abstract 类可有构造方法 10.abstract 类中有 abstract 方法(只声明不实现)和非 abstract 方法 10. interface 声明，只含常量、方法声明，不含方法体，可以声明标量，不可实例化 11.子类不能继承父的初始化方法。当用子类的初始化方法创建一个子类的实例对象时，这个子类声明的所有成员变量都被分配了内存空间，直接父类和所有的祖先类的成员变量也都分配了内存空间。17.如果父类实现了某个接口，则其子类也就自然实现这个接口。18. 非 static 内部类的不可以声明静态变量和静态方法，只可以申请静态常量 20.在类 A 中的内部类 B,若 B 中要使用 B 创建对象，则需要 B obj = new A().new B(); 21 匿名类一定是内部类，可以继承，匿名类不可以声明静态成员变量和静态方法。常用:方法参数（new 类名 or 接口名{重写方法}）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 24.**super**:子类方法通过使用 super 调用父类中被子类隐藏的成员变量和覆盖的成员方法/子类的初始化方法中使用 super 调用父类的初始化方法(必须在子类的初始化方法的第一)25.类方法的方法体中不能有与类的对象有关的内容 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.类方法不能被覆盖。

· 第五章:String 表示一个 UTF-16 格式的 2byte 字符串 2. java.lang.String 包 3.构造函数:**String(char[] a) String(char[] a,startpos,length)** 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 **equals,str.toUpperCase()str.toLowerCase()** 字符串全字符转为大小写 6.**str.startWith(String string) .endWith(String string)**判断 str 是否以 string 开头或者结尾 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 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) 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),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。

· 第六章:泛型 java.util.*;： 1.Java 中集合分为两大类,实现了 Collection 接口(包括 List:ArrayList,LinkedList,Vector 和 Set:HashSet,TreeSet,LinkedHashSet)和实现了 Map 接口(包括 HashMap,TreeMap,HashTable) 2 **ArrayList(int capaticy=16)** 3.ArrayList<E> :**add(E) add(index,E) get(index) Object clone()** 强转 **remove(int i) bool contains(Object) indexOf(Object)(-1) isEmpty() lastIndexOf() set(index,value) removeRange(start,end) size() Object[] toArray()** LinkedList: **addFirst(),getFirst(),removeFirst()**（or Last） 4.Vector 最安全。线程安全版 **List list = Collections.synchronizedList(new ArrayList<>())/或者为 LinkedList<>())**5.HashSet<E>(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(Treeset<String>is=new TreeSet<String>();Itertator<String> 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 结果在 A 中 6.TreeSet<E>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 obj.attr >this.attr?-1:(obj.attr==this.attr?0:1);}升序. String 的比较：标点最前，然后大写字母，然后小写。线程同步 **SortedSet s = Collections.synchronizedSortedSet(new TreeSet<>());**7.HashMap 无序 快 TreeMap 有序 慢 LinkedHashMap 有序 快 7.HashMap<T,E>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<E>的类,方法:**public int compare(E o1,E o2) { return o1.attr>o2.attr?1:(o1.attr==o2.attr?0:-1);}** 线程安全：**Map m=Collections.synchronizedMap(new HashMap(..))**

· 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(String str),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

·第八章:3.没有结束 run()方法之前不能再 start()否则 IllegalThreadStateException 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(唤醒所以等待的线程) 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(this){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。