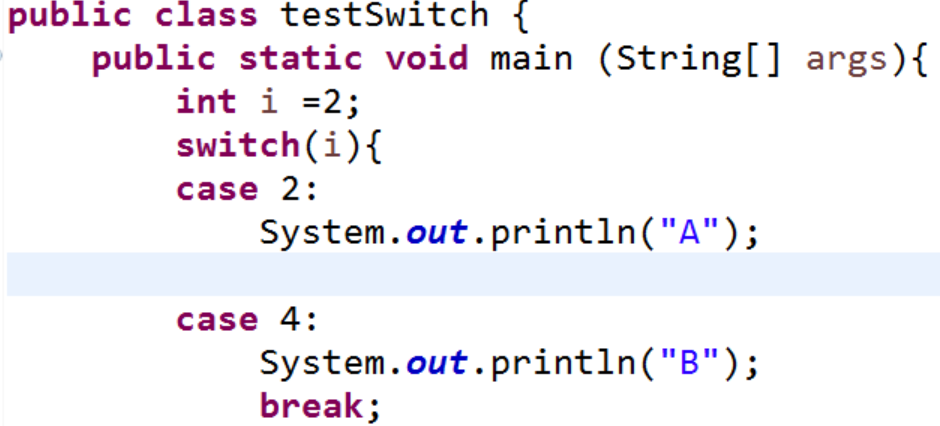
5.case穿透

产生这种现象的原因:由于某个case语句中缺少或者漏写break语句所导致的结果



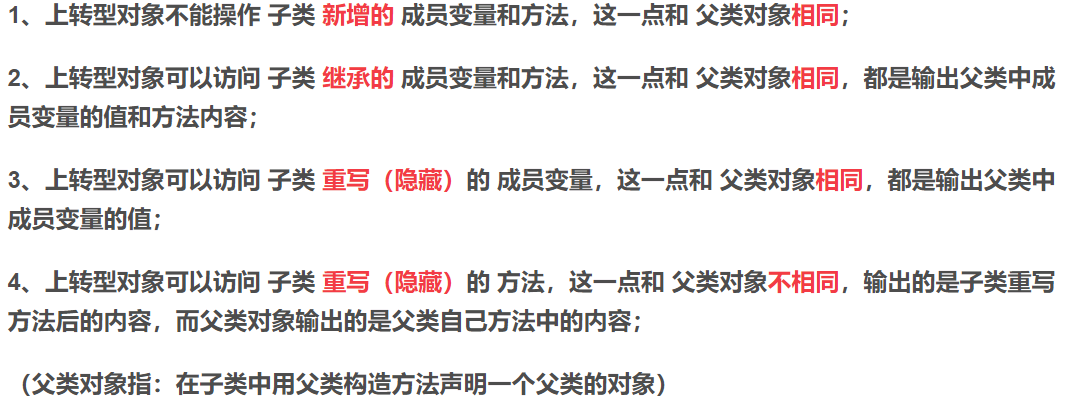
输出结果:执行了case2后由于没有break语句导致继续执行下一条case4语句,然后遇到case4中的break语句后程序结束输出A和B,正常情况应该是只输出A.

1. 上转型对象（多态）

将子类对象的引用赋给父类的对象（也就是使用子类的构造方法创建引用，再赋给父类对象），那么这个父类的对象就是子类对象的上转型对象

父类：A 子类B extends A

A test=new B();

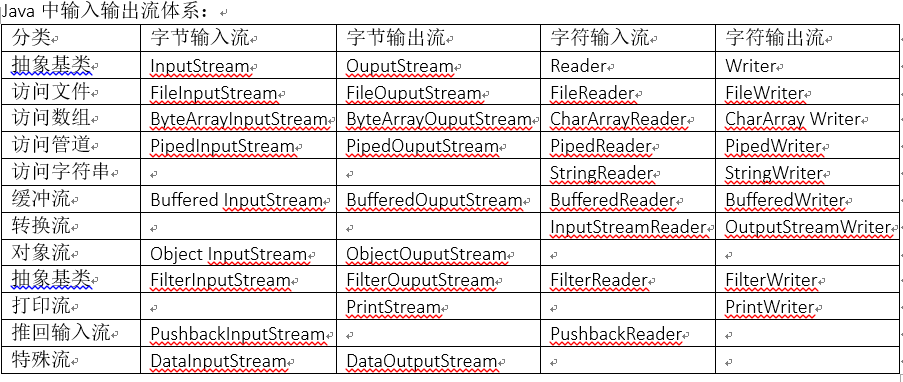


1. 字节字符流

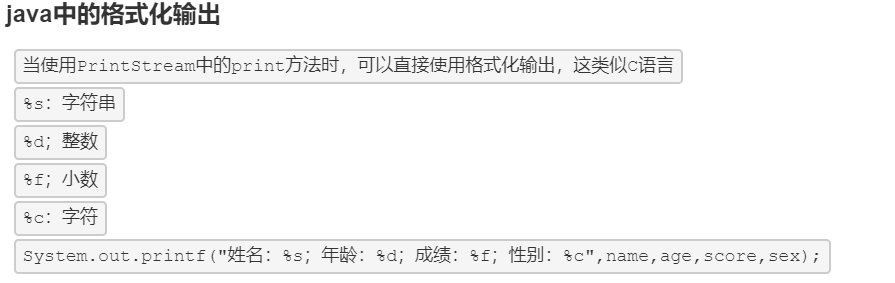
字节流：按字节读取。字符流：按字符读取。

字符流读取方便，字节流功能强大，当不能用字符流时，可以用字节流。

字节流基类：InputStream、OutputStream



13.格式化流



1. File常用方法

（一）访问文件名或路径

1）String getName()  返回File对象所表示的文件名或文件路径

2）String getPath()    返回File对象所对应的相对路径名。

3）File getAbsoluteFile() 返回File对象的绝对路径文件

4）String getAbsolutePath() 返回File对象所对应的绝对路径名

5）String getParent（） 返回File对象所对应目录的父目录

6) boolean renameTo(File dest) 重命名File对象的文件或目录

（二）文件检测

1boolean exists()   判断File对象的文件或目录是否存在

2)bool canWrite()     判断File对象是否可写

3)boolean canRead()判断File对象是否可读

4)boolean isDirectory() 判断File对象是否是目录

5)boolean isFile() 判断File对象是否是文件

6)boolean  isAbsolute() 判断File对象是否采用绝对路径

(三）文件信息

1)long length() ; File对象对应文件的长度

2)long lastNodified()   File对象最后修改的时间

(四）文件操作

1）boolean createNewFile() ;      检查文件是否存在，当文件不存在时创建一个新的文件

2) boolean delete()             删除File对象所对应的文件或目录

(五）目录操作

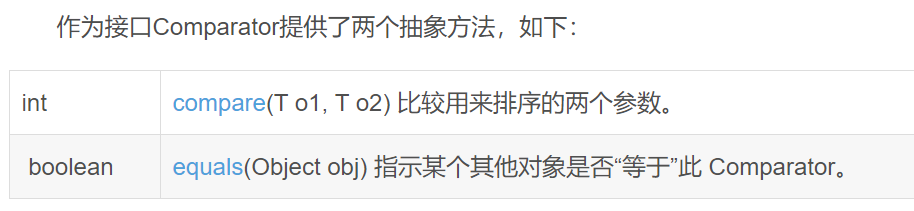
1)boolean mkdir()       创建一个File对象所对应的路径

2)String[] list()         列出File对象所有的子文件名和路径名

3)File[] listFile()     列出File对象的所有子文件或路径

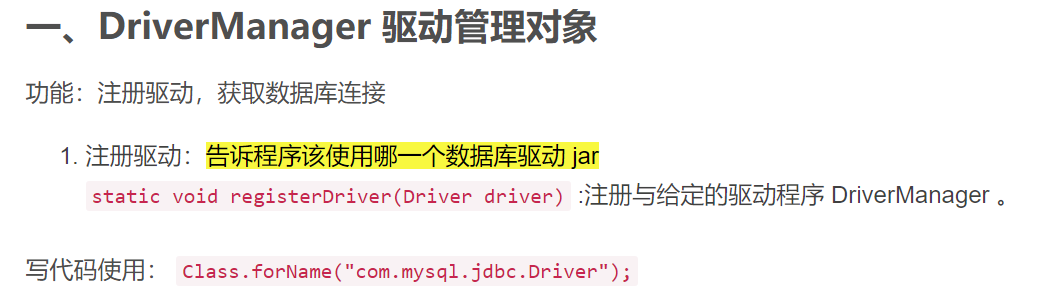
4)static File[] listRoots()   列出系统所有的根路径

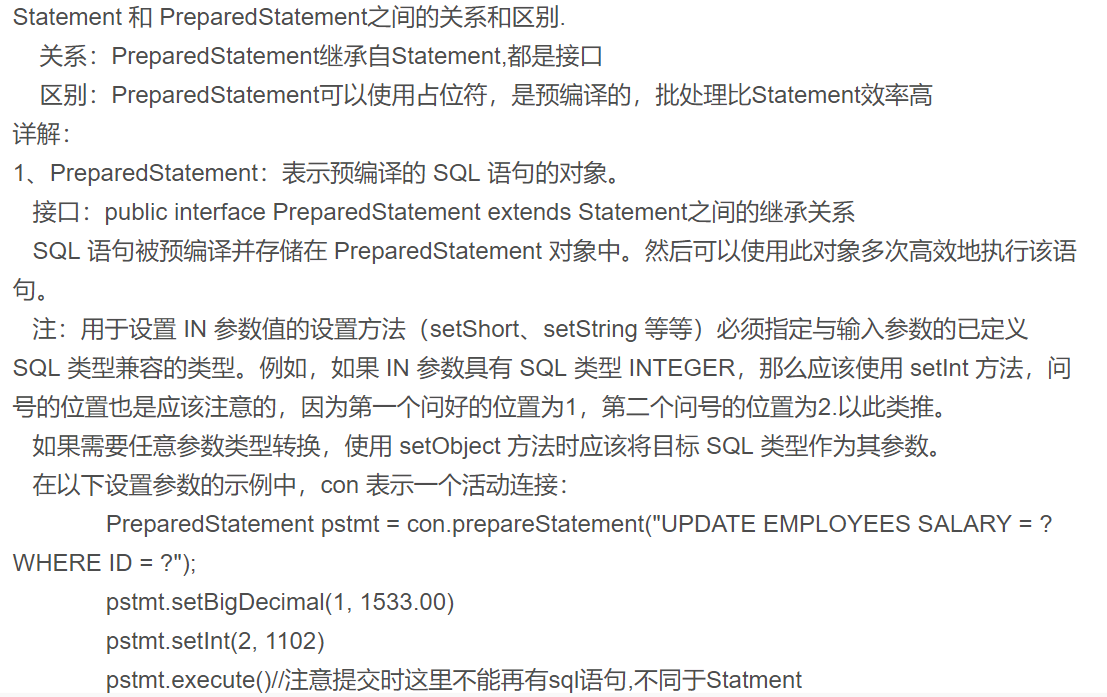
15.Comparator接口

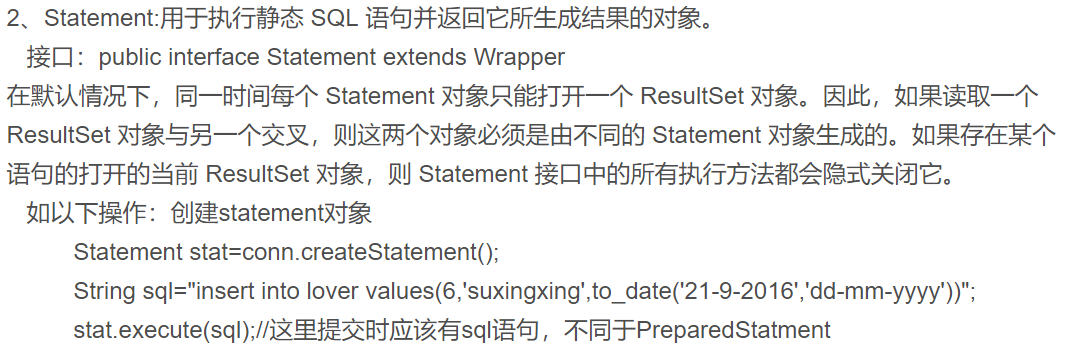


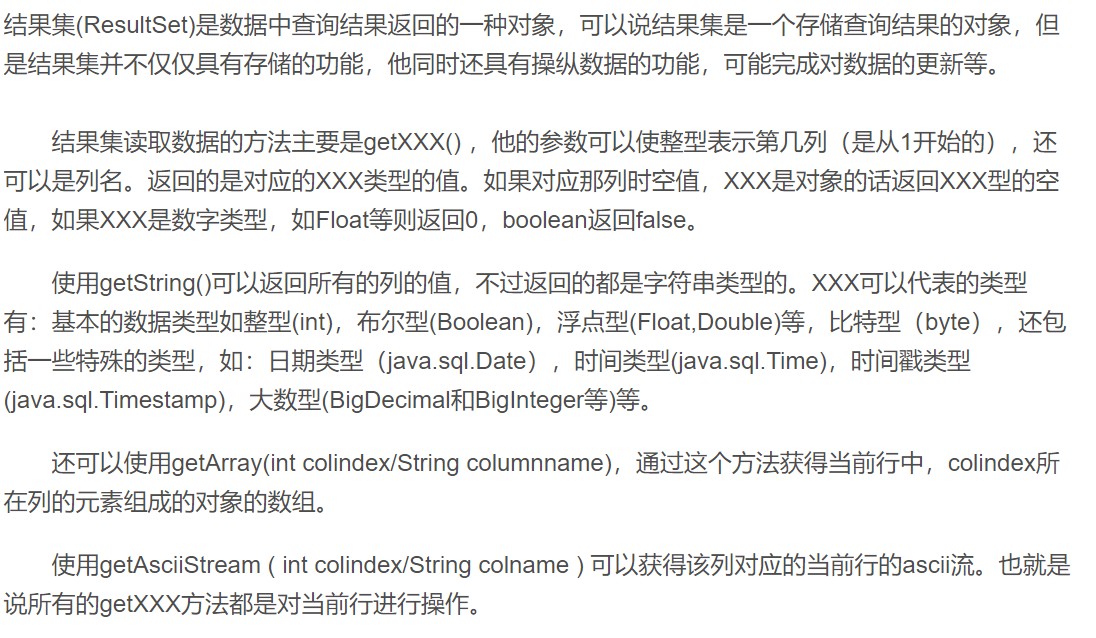
　Compare（）比较用来排序的两个参数。根据第一个参数小于、等于或大于第二个参数分别返回负整数、零或正整数。 在前面的描述中，符号 sgn(expression) 表示 signum 数学函数，根据 expression 的值为负数、0 还是正数，该函数分别返回 -1、0 或 1。

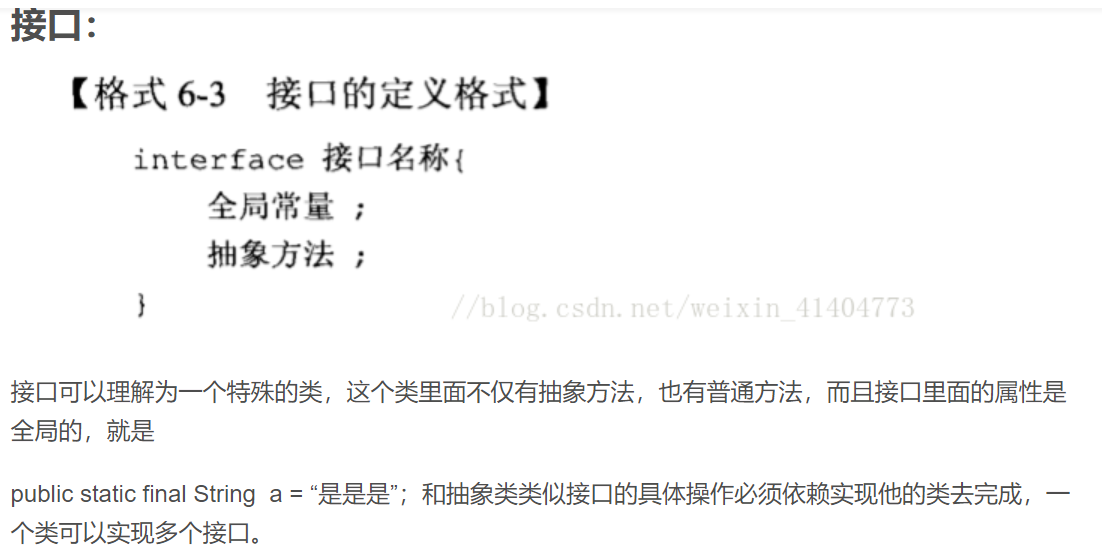
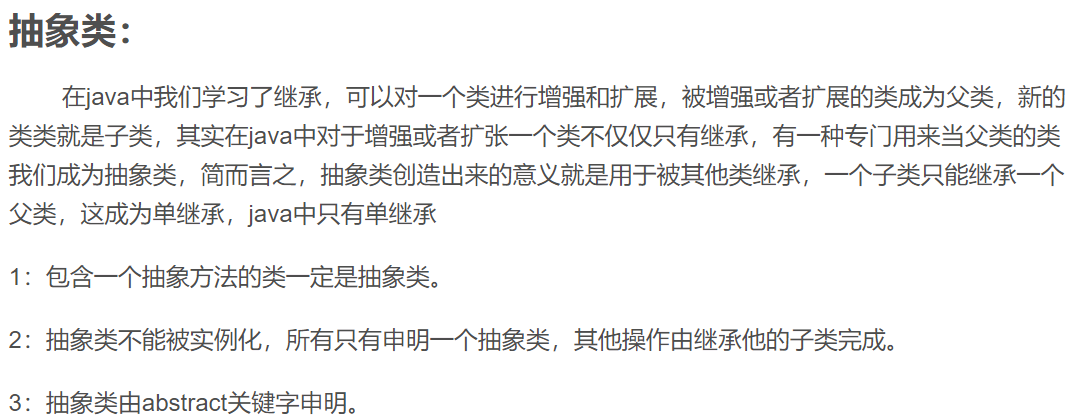
　Equals（）指示某个其他对象是否“等于”此 Comparator。此方法必须遵守 Object.equals(Object) 的常规协定。此外，仅当 指定的对象也是一个 Comparator，并且强行实施与此 Comparator 相同的排序时，此方法才返回 true。因此，comp1.equals(comp2) 意味着对于每个对象引用 o1 和 o2 而言，都存在 sgn(comp1.compare(o1, o2))==sgn(comp2.compare(o1, o2))。

1. JDBC





ResultSet

1. 抽象类接口
2. Sleep wait



sleep()：在指定的毫秒数内让当前"正在执行的线程"休眠(暂停执行)。 这个"正在执行的线程"是指Thread.currentThread()返回的线程。但是sleep不会释放锁。

wait实例方法：使线程进入等待队列。

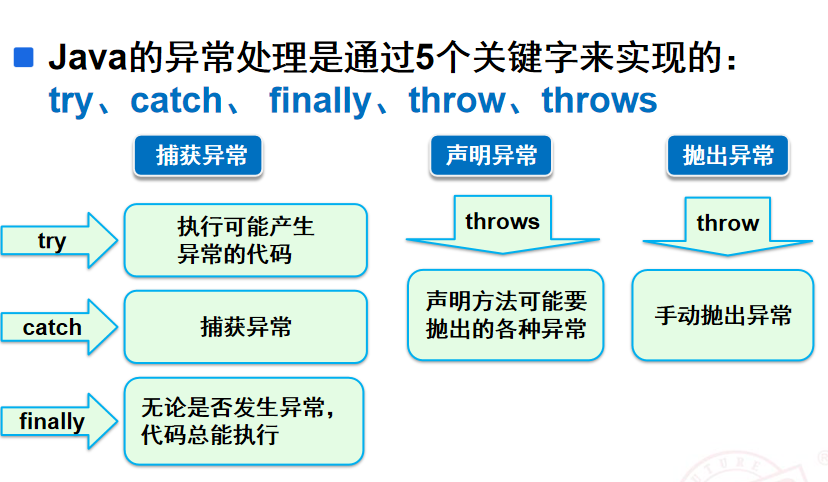
两者最主要的区别在于：sleep 方法没有释放锁，而 wait 方法释放了锁 。

两者都可以暂停线程的执行。

Wait 通常被用于线程间交互/通信，sleep 通常被用于暂停执行。

wait() 方法被调用后，线程不会自动苏醒，需要别的线程调用同一个对象上的 notify() 或者 notifyAll() 方法。sleep() 方法执行完成后，线程会自动苏醒。或者可以使用 wait(long timeout)超时后线程会自动苏醒。

21异常



1. 线程

线程的生命周期：新建、就绪、运行、阻塞、死亡。

一个线程不是被创建了马上就开始执行，也不是一直处于执行状态。

就绪  
当线程对象调用start()方法之后，该线程处于就绪状态。JVM会为其创建方法调用栈和程序计数器，等待调度运行。

运行  
如果处于就绪状态的线程获得了CPU，开始执行run()方法的线程执行体，则该线程处于运行状态。

阻塞  
阻塞状态是指线程因为某种原因放弃了CPU使用权，暂时停止运行。  
阻塞分三种情况：  
（1）等待阻塞（o.wait->等待队列）：运行（running）的线程执行o.wait()方法，JVM会把该线程放入等待队列（waiting queue）中；  
（2）同步阻塞（lock->锁池）：运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池（lock pool）中；  
（3）其他阻塞（sleep/join）:运行的线程执行Thread.sleep(long ms)或thread.join方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时，join()等待线程终止或超时，或者I/O处理完毕时，线程重新转入可运行状态中。

销毁  
如果线程正常执行完毕后或线程被提前强制性的终止或出现异常导致结束，那么线程就会被销毁，释放资源。

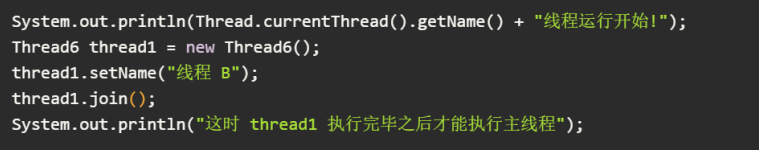
线程的常用方法

wait  
作用：强迫一个线程等待，线程进入WAITING状态，只用等待另外线程的通知或被中断才会返回；  
注意：调用wait()方法后，会释放对象的锁。因此，wait方法一般用在同步方法或同步代码块中。

Sleep  
作用：强迫一个线程睡眠N毫秒，线程进入TIMED\_WAITING状态，只有等待另外线程的通知或者被中断才会返回；  
注意：与wait方法不同的是sleep不会释放当前占有的锁

yield  
作用：使当前线程让出CPU执行时间片，与其他线程一起重新竞争CPU时间片；  
注意：一般情况下，优先级高的线程有更大的可能性成功竞争得到CPU时间片，但这不是绝对的，有的操作系统对线程优先并不敏感。

interrupt  
作用：中断一个线程，其本意是给这个线程一个通知喜好，会影响这个线程内部的一个中断标识位；  
注意：  
（1）interrupt不会改变线程的状态，既不会强制线程进入阻塞，终止等状态；  
（2）若线程处于TIMED-WAITING状态，这时调用interrupt方法，会抛出InterruptException，从而使线程提前结束TIMED-WAITING状态；  
（3）许多申明抛出InterruptException的方法，抛出异常前，都会清除中断标识位，所以抛出异常后，调用isInterrupt方法会返回false;  
（4）可以调用thread.interrupt方法，在线程的run方法内部可以根据thread。isInterrupt的值来优雅的终止线程。

join  
作用：使当前线程转为阻塞状态，直到另一个线程结束，当前线程再由阻塞状态变为就绪状态；  
使用场景：主线程中启动子线程，主线程等待子线程的返回状态。  


notify  
作用：Object类中的notify方法，唤醒在此对象监视器上等待的单个线程；  
TIP:  
如果所有线程都在此对象上等待，则会选择唤醒其中一个线程，选择是任意的，并在唤醒该线程前，调用该线程的wait()方法，在对象的监视器上等待，直到当前的线程放弃此对象上的锁定，才能继续执行被唤醒的线程，被唤醒的线程将以常规方式与在该对象上主动同步的其他所有线程进行竞争。类似方法还有notifyAll，唤醒在此监视器上等待的所有线程。

synchronized 关键字

代表这个方法加锁,相当于不管哪一个线程(例如线程A)，运行到这个方法时,都要检查有没有其它线程B(或者C、 D等)正在用这个方法(或者该类的其他同步方法)，有的话要等正在使用synchronized方法的线程B(或者C 、D)运行完这个方法后再运行此线程A,没有的话,锁定调用者,然后直接运行。它包括两种用法:synchronized 方法和 synchronized 块。

1. 字符串常用方法

//获取字符串长度

int strLenth=str.length();

//获取字符串指定位置的字符

char indexChar=str.charAt(0);

//查找某个字符在字符串中首次出现的位置

int firstIndex=str.indexOf("1");

//查找某个字符在字符串中最后出现的位置

int lastIndex=str.lastIndexOf("1");

//去除字符串中的首尾空格

String newStr=str.trim();

//截取字符串

String subStr=str.substring(0,2);

//字符串替换

String reStr=str.replace("1", "4");

//替换第一次出现的字符

String firReStr=str.replaceFirst("1", "4");

//判断字符串的开始和结尾

boolean startWith=str.startsWith("我");

boolean endWith=str.endsWith("3");

//判断两个字符串是否相等

boolean equals=str.equals("");

//忽略大小写判断两个字符串是否相等

boolean equalsIgnoreCase=str.equalsIgnoreCase("AAA");

//判断某个字符串是否包含另一个字符串

boolean contains=str.contains("");

//将字母全部转换成大写

String lowerStr=str.toLowerCase();

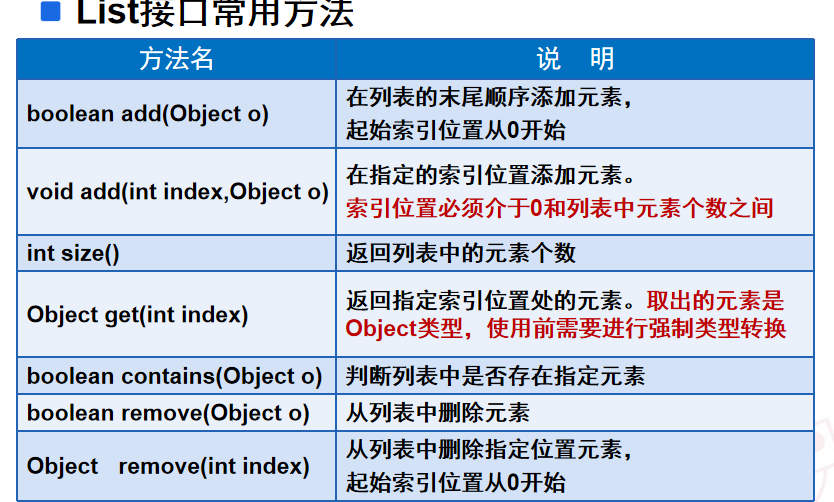
//将字母全部转换成小写

String upperCase=str.toUpperCase();

//将字符串分割,多个分隔符可用|隔开，例如下面这个是按照“。”和“,”分割的

String splite[]=str.split("。|,");

1. 集合框架



25.时间

获得当前时间

       Date date=new Date();

       //当前日期的毫秒值

       System.out.println(date.getTime()); // 1533547522877

       //获得当前日期的毫秒值

    System.out.println(System.currentTimeMillis()); // 1533547522877

日期格式化

日期转字符串

y 代表年

M 代表月

d 代表日

E 代表星期

H 代表24进制的小时

h 代表12进制的小时

m 代表分钟

s 代表秒

S 代表毫秒

public class DateTest {

    public static void main(String[] args) {

       SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

       SimpleDateFormat sdf1=new SimpleDateFormat("yyyy-MM-dd E HH:mm:ss SSS");

       Date date=new Date();

       String dateStr=sdf.format(date);

       System.out.println(dateStr);

       System.out.println(sdf1.format(date));

    }

编程1 封装继承抽象类多态重写

public class Student extends Person {

    public Student (String name, String age) {

        super.setName(name);

        super.setAge(age);

    }

    @Override

    public void Study() {

        System.out.println("姓名："+super.getName()+" 年龄："+super.getAge()+" 学习Java");

    }

}

class Studenttest{

    public static void main(String[] args) {

        Person p=new Student("帅哥", "20");

        p.Study();

    }

}

public abstract class Person {

    private String name;

    private String age;

    public abstract void Study();

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getAge() {

        return age;

    }

    public void setAge(String age) {

        this.age = age;

    }

}

编程2 JDBC

// 1、创建驱动获取连接

Class.forName("com.mysql.jdbc.Driver");

con=DriverManager.getConnection("jdbc:mysql://127.0.0.1/jsd","root","1234");

// 2、创建语句对象

st=con.createStatement();

// 3、执行SQL语句

String sql="select \* from abc";

Resultset rs=st.executeQuery(sql);

// return rs;

sql="insert into abc(name) values(‘1’) ";

rs=st.executeUpdate(sql);

return rs;

编程3 线程同步 字节复制图片

import java.io.\*;

import java.util.Scanner;

public class FilePic {

    public static void main(String[] args) {

        PicThread pt=new PicThread();

        pt.start();

        System.out.println("是否开始复制图片？");

        Scanner sc=new Scanner(System.in);

        if("1".equals(sc.nextLine()))

        {

            synchronized(pt){

                  pt.notify();

            }

        }

        sc.close();

    }

}

public class PicThread extends Thread {

    public void run()

    {

        try {

            synchronized(this){

                this.wait();

            }

            FZ();

        }

        catch (InterruptedException e) {

        }

    }

    public synchronized void FZ()

    {

        String filename1 = "D:/1.jpg";

        String filename2 = "E:/1.jpg";

        try{

            FileInputStream fis=new FileInputStream(filename1);

            FileOutputStream fos=new FileOutputStream(filename2);

            byte[] words = new byte[1024];

            int count=0;

            while ((count=fis.read(words))!= -1) {

                fos.write(words, 0, words.length);;

            }

            System.out.println("复制完成");

            fos.close();

            fis.close();

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

编程4 集合

Set:

Set<String> set = new HashSet<String>();

        set.add("one");

        set.add("two");

        set.add("three");

        for(String str : set){

            System.out.println(str);

        }

List Map

public class Studenthash<T> {

    public static void main(String[] args) {

        stu x1=new stu(1,"许欣");

        stu x2=new stu(2,"邹翔");

        stu x3=new stu(3,"张方泷");

        Map<String,List<stu>> stumap=new HashMap<>();

        List<stu> s1=new ArrayList<>();

        List<stu> s2=new LinkedList<>();

        s1.add(x1);

        s2.add(x2);

        s1.add(x3);

        stumap.put("普通班", s1);

        stumap.put("强化班", s2);

        ///foreach

        for (String key : stumap.keySet()) {

            for(int i=0;i<stumap.get(key).size();i++)

            {

                System.out.print(key + ",学生：");

                System.out.println(((stu)stumap.get(key).get(i)).toString());

            }

        }

        ///迭代

        Set set=stumap.keySet();

        Iterator<String> it = set.iterator();

        while (it.hasNext()) {

            String n = it.next();

            for(int i=0;i<stumap.get(n).size();i++)

            {

                System.out.print(n + ",学生：");

                System.out.println(((stu)stumap.get(n).get(i)).toString());

            }

        }

    }

}

class stu{

    int id;

    String name;

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

    public String toString()

    {

        return "学号："+getId()+" 姓名："+getName();

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public stu(int id, String name) {

        this.id = id;

        this.name = name;

    }

}

编程5 IO字符流 读写文件

import java.io.FileWriter;

import java.util.Scanner;

import java.io.\*;

public class Txt {

    public static void main(String[] args) {

        try {

        System.out.println("请输入要写入的内容：");

        Scanner sc=new Scanner(System.in);

        String txt=sc.nextLine();

        FileWriter fw= new FileWriter("D:/啧啧啧.txt", true);

        fw.write(txt);

        fw.flush();

        fw.close();

        System.out.println("写入成功！");

        sc.close();

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        } catch (IOException e) {

              e.printStackTrace();

        }

    }

}