# Day21

今日主要内容：

1. 打印流
2. 标准输入/输出流
3. 异常处理问题
4. IO流资源释放方式
5. 程序、进程、线程三个概念
6. 并行和并发
7. 两种线程的实现方式
8. 线程中常的方法

## 打印流

PrintStream/printWriter

这两个类都是打印流，printStream是字节打印流，printWriter就字符打印流,它们都拥用各自己父类中的读和写的方法。我们关注的是不是writer方法。关注的是两个特殊方法。

print() /println()

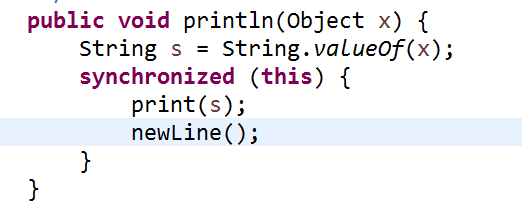
### PrintStream

他是一个字节打印流

PrintStream对象的获取方式:

1. System.out获取一个PrintWriter对象，他们输出的目的地是控制台

关于输出数组数据的问题



|  |
| --- |
| public class PrintStreamTest {    public static void main(String[] args) {      System.out.println(1+2);    int[] a= {1,2,3,4,5,6};  System.out.println(a);    char[] b= {'a','b','c','d'};  System.out.println(b);    }  } |

Print 和println(), 把打印的各种数据类型的数据都要变成一个字符串，然后进行输出

如果打印的是char的数组的话，输出的是各元素的值（内容）。

对于他们数组和其它对象来讲打印的都是调用了toString()方式所返回的字符串的值

1. 还可以通过PrintStream的构造方法来创建对象

PrintStream(File file)

PrintStream(String path)

PrintStream(OutputStream out) //它可以把一个节流输出流，来包装一个PrintWriter,这样就可以使用print和println() 输出数据了

|  |
| --- |
| public static void PrintStream\_default() throws FileNotFoundException {    FileOutputStream fos=new FileOutputStream("aaa.txt");  PrintStream ps=new PrintStream(fos);  ps.println("http://www.baidu.com");  ps.close();    } |

## PrintWriter

PrintWriter是一个字符打印流。在内存中就会有一个缓存冲，该类提供了自动刷新的方法。

1. 构造方法必须提供自动刷新的设置

**[PrintWriter](mk:@MSITStore:C:\\Users\\ibm\\Desktop\\JDK_API_1.6_zh_中文.CHM::/java/io/../../java/io/PrintWriter.html" \l "PrintWriter(java.io.OutputStream, boolean))**([OutputStream](mk:@MSITStore:C:\\Users\\ibm\\Desktop\\JDK_API_1.6_zh_中文.CHM::/java/io/../../java/io/OutputStream.html" \o "java.io 中的类) out, boolean autoFlush)

**[PrintWriter](mk:@MSITStore:C:\\Users\\ibm\\Desktop\\JDK_API_1.6_zh_中文.CHM::/java/io/../../java/io/PrintWriter.html" \l "PrintWriter(java.io.Writer, boolean))**([Writer](mk:@MSITStore:C:\\Users\\ibm\\Desktop\\JDK_API_1.6_zh_中文.CHM::/java/io/../../java/io/Writer.html" \o "java.io 中的类) out, boolean autoFlush)

1. 调用了自带刷新的方法 println() printf() format()

|  |
| --- |
| public static void main(String[] args) throws IOException {  //调用构造方法，该构种过方法能设置自动刷新参数  FileWriter fw=new FileWriter("bbb.txt");  PrintWriter pw=new PrintWriter(fw,true);  pw.write(21);  pw.write("zhangsan");  pw.print(true);  pw.println("lisi");//强制刷新缓冲区  pw.close();  } |

## 标准的输入/输出流

System.in //从键盘上接收数据 类型是字节输入流

更改标准的输入设备： setIn(InputStream)

System.out //输出到控制台， 类型PrintStream

更改标准的输出设备： setOut(PrintStream)

## Properties

Properties也是一个基于Key-Value 属性集，它的Key和Value，都是字符串。

Properties 类表示了一个持久的属性集。

持久：就是把数据保存到一个文件中。

|  |
| --- |
| public static void putPropertiesData() {    //创建一个属性列表对象  Properties pp=new Properties();  pp.put("className","cn.offcn.Person");  pp.put("name","zhangsan");  pp.put("age",22);    Set<Map.Entry<Object,Object>> sets=pp.entrySet();    for(Map.Entry<Object,Object> me : sets) {  System.out.println(me.getKey()+"="+me.getValue());  }        } |

### Properties提供了一些特有的方法：

**[setProperty](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Properties.html" \l "setProperty(java.lang.String, java.lang.String))**([String](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/lang/String.html" \o "java.lang 中的类) key, [String](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/lang/String.html" \o "java.lang 中的类) value)

**[getProperty](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Properties.html" \l "getProperty(java.lang.String))**([String](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/lang/String.html" \o "java.lang 中的类) key)

**[getProperty](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Properties.html" \l "getProperty(java.lang.String, java.lang.String))**([String](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/lang/String.html" \o "java.lang 中的类) key, [String](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/lang/String.html" \o "java.lang 中的类) defaultValue)

**[stringPropertyNames](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Properties.html" \l "stringPropertyNames())**() //属性列表中的key的所有的集合形式

### 可利用properties对象从文件中读取数据

**[load](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Properties.html" \l "load(java.io.InputStream))**([InputStream](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/io/InputStream.html" \o "java.io 中的类) inStream)

**[load](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Properties.html" \l "load(java.io.Reader))**([Reader](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/io/Reader.html" \o "java.io 中的类) reader)

1.要有一个属性文件 xxx.properties

|  |
| --- |
| bookName=JavaBase  booPrice=99.9  bookAuthor=lisi |

代码:

|  |
| --- |
| public static void loadPropertiesData() throws Exception {  Properties pp=new Properties();  FileReader reader=new FileReader("aaa.properties");  pp.load(reader);    //设置属性值  pp.setProperty("name", "wangwu");  pp.setProperty("bookAuthor", "zhaoliu");              //遍历属性列表  Set<String> names=pp.stringPropertyNames();  for(String name : names) {  System.out.println(name+"="+pp.getProperty(name));  }    //把数据同步到文件中    } |

### 把内容中的properties对象中的属性和值同步到文件中（持久化）

**[store](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Properties.html" \l "store(java.io.OutputStream, java.lang.String))**([OutputStream](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/io/OutputStream.html" \o "java.io 中的类) out, [String](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/lang/String.html" \o "java.lang 中的类) comments)

**[store](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/util/Properties.html" \l "store(java.io.Writer, java.lang.String))**([Writer](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/io/Writer.html" \o "java.io 中的类) writer, [String](mk:@MSITStore:C:\\Users\\Administrator\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/../../java/lang/String.html" \o "java.lang 中的类) comments)

修改了内容中的属性列表

//设置属性值

pp.setProperty("name", "wangwu");

pp.setProperty("bookAuthor", "zhaoliu");

//把数据同步到文件中

FileWriter fw=new FileWriter("aaa.properties");

pp.store(fw,"add name attribute,update bookAuthor is zhaoliu");

在以后阶段中，经常使用properties属性文件，来做一些配置，这样解决我程序中硬编码的问题。

什么是硬编码：在程序写死的数据，如要需要更改数据的话，需要打开源码进行更改，重新把.Java文件编译.class文件，才能生效。

软编码：程序所用的一些数据是在配置文件中存储的，如果程中需要此数据，可以使用Properites从文件中读取，当修改properties属性文件中某一部key的value时，我们的程序不用重新编译。

### 异常处理问题

1. 编译时异常如果该异常你能够处理，可以使用try{}catch(){}来处理。处理不了就直接把异常对象向上抛，必须配合throws使用，抛给调用者。
2. 对于运行时异常可以用try{}catch(){}来获取异常，如果处理不了，throw e;

## 对象流关闭的问题

我们要加try{}catch(){}finally{} 在finally中关闭流资源，如果在finally中关闭了多个流的资源，如果第一流关闭时失败了，后面的流的闭后就停止，导入流不能被关闭。

解决方案： 1.7之前采用finally{}来关闭流，如果在finally{}关闭多个流时，每个流关闭时都要使用try{}catch(){}finally{try{}catch(){}finally{try{}catch(){}finally{}}}

|  |
| --- |
| public static void aaa() throws Exception{  FileReader fr=null;  FileWriter fw=null;  try {  fr=new FileReader("ddd.txt");  int c=0;  while((c=fr.read())!=-1) {  //。。。。  }    fw=new FileWriter("eee.txt");  fw.write('a');    }catch(Exception e) {    }finally {    try {  if(fr!=null) {  fr.close();  }  }finally {  if(fw!=null) {  fw.close();  }  }    }  } |

1.8版本以后，自动关闭流。

|  |
| --- |
| public static void bbb() throws Exception {  try(  FileReader fr=new FileReader("ddd.txt");  FileWriter fw=new FileWriter("eee.txt");    ){    fr.read();  fw.write('c');    }    } |

## 程序、进程、线程三个概念

程序：是以文件形式存储在磁盘上。是一个静态方式

进程：进程（Process）是计算机中的程序关于某数据集合上的一次运行活动，是系统进行资源分配和调度的基本单位。

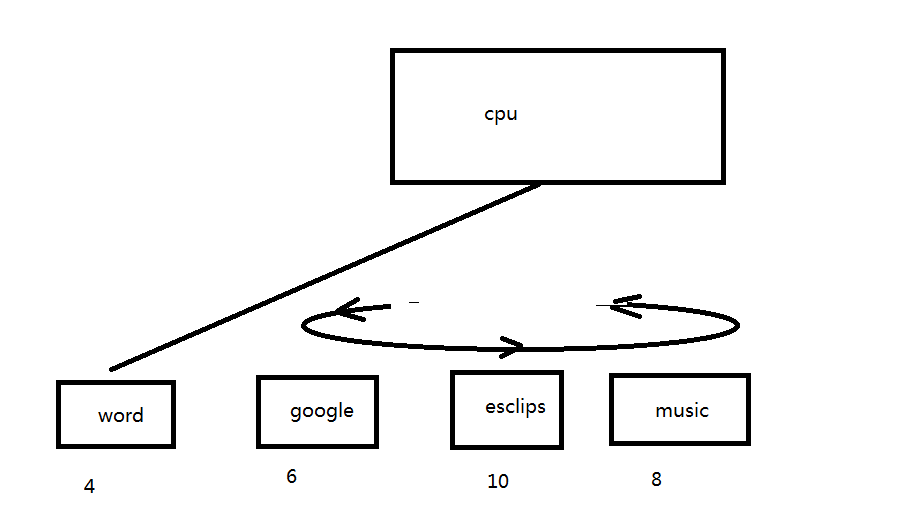
线程：是进程的一个单一控制流程，是进程执行的最小单位。一般情况下，一个进程可以分配好多任务，每个任务就是一个线程。

## 并行与并发

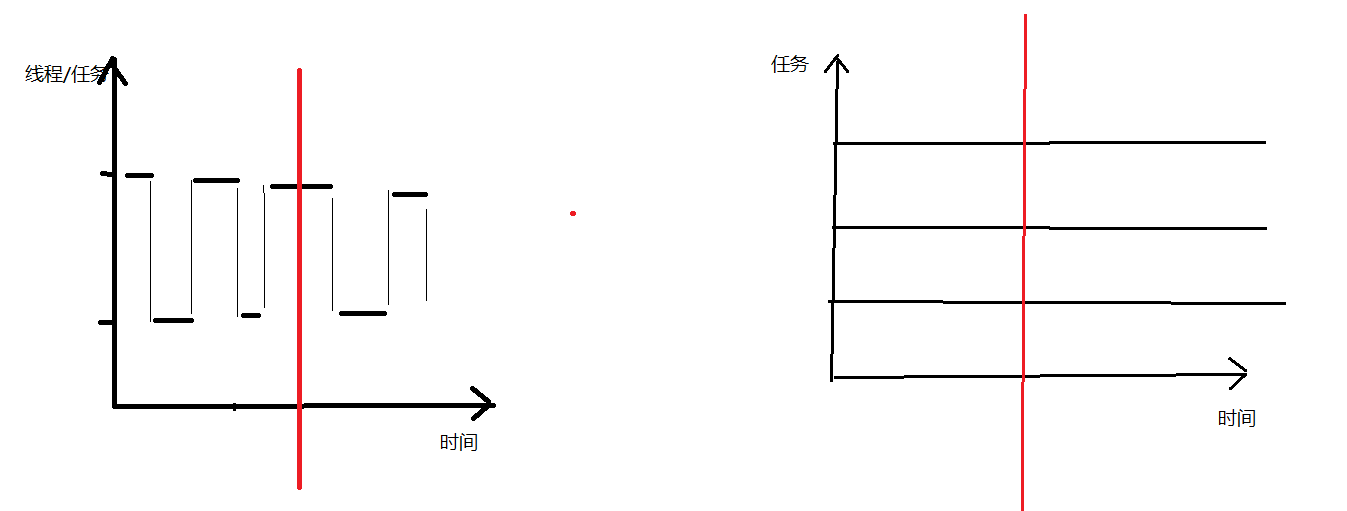
并行： 多个任务要同时运行（有多个CPU）,每个任务都运行在不同的Cpu上。并行编程。

并发： 多个任务同时发起请求，（cpu） 在某一个时刻，只一个任务（线程）在cpu中执行。

Cpu并发执行线程图



并发与并行图：



在某一个时刻只有一个交点，就是并发，如果有多个交点就是并行。

并发提升了Cpu的利用率。对于线程来讲，效率降低了。