* 文件操作File

File构造器

1. 创建File对象

java.io.File用于表示文件（目录），也就是说程序员可以通过File类在程序中操作硬盘上的文件和目录。

File类只用于表示文件（目录）的信息（名称、大小等），换句话说只能访问文件或目录的相关属性，不能对文件的内容进行访问。

创建File对象可以代表文件或者文件夹

1. new File(文件路径名)

通过将给定路径名字符串转换成抽象路径名来创建一个新 File 实例

提示:抽象路径应尽量使用相对路径，并且目录的层级分隔符不要直接写”/”或”\”,应使用File.separator这个常量表示，以避免不同系统带来的差异。

2)new File(父文件夹，文件名)File(File parent,String child)

根据 parent抽象路径名和child路径名字符串创建一个新File实例。

实例：

//创建file对象代表文件夹

File dir=new File("test");

//创建file对象代表 文件夹中的文件

File file=new File(dir, "demo.txt");

//如上file代表 test文件夹中的demo.txt文件

boolean b = file.exists();

System.out.println(b); //true

2.删除文件 API 方法

File的delete方法用于删除此抽象路径名表示的文件或目录。 其方法定义[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE01/DAY05/SUPERDOC/01/index.html):boolean delete()boolean delete()

返回值:当且仅当成功删除文件或目录时，返回 true；否则返回 false。

需要注意的是，若此File对象所表示的是一个目录时，在删除时需要保证此为空目录才可以成功删除(目录中不能含有任何子项)。

实例：

//创建file 对象，代表硬盘上的文件

File file=new File("demo/test.txt");

//有了file对象后就可以利用File类

//提供的API方法操作 demo/test.txt文件

//delete：删除file所代表的硬盘文件

boolean success = file.delete();

System.out.println(success);

//当删除成功时候返回 true，

//删除失败时候返回false

success = file.delete();

System.out.println(success);

3.检查文件是否存在

new File（） 用于创建内存对象，并不是创建文件或者文件夹，甚至并不一定有对应的磁盘文件，Java File 提供了exists用于

检查对应的磁盘文件（文件夹）是否存在：

File file=new File("demo/hello.txt");

//exists: 检查file对象代表的硬盘文件

//是否存在，如果存在则true，否则false

boolean b = file.exists();

System.out.println(b);//true

file.delete();

//删除以后磁盘文件就不存在了，但是内存对象还在

b = file.exists();

System.out.println(b);//false

4.文件分隔符

\* 文件分隔符问题：

\* 1. Windows 的文件分隔符 ：\

\* - D:\demo\test.txt

\* - demo\test.txt

\* - File.separator = \

\* 2. Linux 的文件分隔符 ：/

\* - /home/soft01/demo/test.txt

\* - demo/test.txt

\* - File.separator = /

\* 3. File 类提供了自动适应操作系统的文件分隔符

\* 变量File.separator或随着操作系统自动变化，

\* 可以利用这变量，编写跨系统的程序：

\* - "demo"+File.separator+"test.txt"

\* 4. 当使用 “/” 时候 Java 会自动使用所有操作

\* 系统，这样更加方便！

public class Demo{

public static void main(String[] args) {

String path= "demo"+File.separator+"test.txt";

File file = new File(path);

boolean b = file.exists();

System.out.println(b);//false

}

5.File API 提供了读取文件（文件夹）属性的方法

* File的isFile方法用于判断当前File对象表示的是否为一个文件

**1)boolean isFile()boolean isFile()**

该方法若返回true,这表示File表示的是一个文件。

* File的length方法用于返回由此抽象路径名表示的文件的长度，其定义为:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE01/DAY05/SUPERDOC/01/index.html)

**2)long length() long length()**

该方法返回的long值表示该文件所占用的字节量。

* File的isDirectory方法用于判断当前File对象表示的是否为一个目录

**3)boolean isDirectory()boolean isDirectory()**

返回值:若File对象表示的是一个目录，则返回true

public class Demo{

public static void main(String[] args) {

File dir=new File("test");

File file=new File(dir, "demo.txt");

//dir代表一个文件夹

//file 代表一个文件

//检查是否是文件

System.out.println(dir.isFile());

System.out.println(file.isFile());

//检查文件的长度

long length = file.length();

System.out.println(length);

//检查文件的最后修改时间

long time = file.lastModified();

Date date = new Date(time);

System.out.println(date);

//检查文件的读写属性

boolean read=file.canRead();

boolean write=file.canWrite();

boolean hidden = file.isHidden();

System.out.println(read);

System.out.println(write);

System.out.println(hidden);

6.创建文件夹

* mkdir方法

File的mkdir方法用于创建此抽象路径名指定的目录。其方法定义:[copytext](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE01/DAY05/SUPERDOC/01/index.html)pop-upboolean mkdir()boolean mkdir()

返回值：当且仅当已创建目录时，返回 true；否则返回 false

* mkdirs方法

File的mkdirs方法用于创建此抽象路径名指定的目录，包括所有必需但不存在的父目录。注意，此操作失败时也可能已经成功地创建了一部分必需的父目录。其方法定义:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE01/DAY05/SUPERDOC/01/index.html)boolean mkdirs()

boolean mkdirs()

返回值：当且仅当已创建目录以及所有必需的父目录时，返回 true；否则返回 false

* createNewFile方法

createNewFile: 创建新文件

如果创建成功返回true，创建失败返回false，如果没有写入权限，则出现异常！！

File file = new File("abc");

//mkdir()用于创建文件夹，如果成功

//创建文件夹，则返回true，否则false

boolean b = file.mkdir();

System.out.println(b);//true

//创建系列文件夹

File file2=new File("def/demo/test");

//mkdirs 用于创建一系列父子文件夹

b = file2.mkdirs();

System.out.println(b);

//使用绝对路径创建文件夹

File file3=new File("d:/demo/test");

//new File("/home/soft01/demo/test");

file3.mkdirs();

创建文件

File file = new File("test/abc.txt");

//调用file对象的API方法

boolean b = file.createNewFile();

System.out.println(b);//true

//使用绝对路径创建文件

7.文件（文件夹）改名

File file1=new File("test/abc.txt");

File file2=new File("test/def.txt");

//file1 是存在的文件，

//file2 是一个不存在的文件

//将file1的名字改名为file2对应的名字

boolean b = file1.renameTo(file2);

//改名之后：file1就不存在了（abc.txt）

// file2存在(def.txt)

System.out.println(b);//true

}

}

8.列出文件夹的内容

listFiles方法用于返回一个抽象路径名数组，这些路径名表示此抽象路径名表示的目录中的文件。其方法定义:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE01/DAY05/SUPERDOC/01/index.html)

File[] listFiles()

返回值：抽象路径名数组，这些路径名表示此抽象路径名表示的目录中的文件和目录。如果目录为空，那么数组也将为空。如果抽象路径名不表示一个目录，或者发生 I/O 错误，则返回 null。

//创建file对象代表test文件夹

// new File("D:/");

// new File("/home/soft01/");

File dir=new File("test");

//查询 test文件夹的内容列表

//返回值是File类型的数组，有的是文件有的是文件夹

File[] files=dir.listFiles();

//在控制台输出文件夹中全部的内容

for(File file:files){

//文件夹显示为： [文件夹名]

if(file.isDirectory()){

System.out.println("["+file.getName()+"]");

}else{

System.out.println(file.getName());

}

9.递归删除文件夹的内容

如何删除一个包含文件的文件夹

public static boolean delete(File dir){

//删除一个文件夹的步骤：

//1. 列出文件夹的全部内容

//2. 遍历每个文件，并且调用文件的delete()

//3. 删除文件夹

//4. 返回删除的结果

if(! dir.exists()){ return false;}

File[] files=dir.listFiles();

for(File file:files){

if(file.isDirectory()){

//删除子文件夹

//删除子文件的算法与删除当前

//文件夹的算法是一致的。利用

//递归删除子文件夹

delete(file);

}else{

file.delete();

}

}

return dir.delete();

}

10.递归统计文件夹的大小

统计一个文件夹中全部文件的总长度

//统计dir对应的文件夹中文件的总大小

public static long count(File dir){

//1. 查找dir的全部内容

//2. 遍历每个文件，累加文件的大小

//3. 返回统计结果

if(! dir.exists()){ return 0;}

if(dir.isFile()){return dir.length();}

File[] files=dir.listFiles();

long sum = 0;

for(File file:files){

if(file.isDirectory()){

//统计子文件夹的总大小:统计子文件夹时候和统计当前文件夹的算法是一样的！

long l = count(file);

sum += l;

}else{

//统计一个文件

sum+=file.length();

}

}

return sum;

}

}

11.有条件列目录

* **FileFilter接口**

通过listFiles方法我们可以获取一个目录下的所有子项，但有些时候我们并不希望获取全部子项，而是想获取部分满足我们实际需求的子项时，我们可以使用File的重载方法:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE01/DAY05/SUPERDOC/01/index.html)File[] listFiles(FileFilter filter)

File[] listFiles(FileFilter filter)

这里我们看到，该重载方法 要求我们传入一个参数，其类型是FileFilter。什么是FileFilter呢? FileFilter是用于抽象路径名的过滤器，说白了就是定义一个规律规则，那么结合listFiles方法，我们就可以将满足此过滤规则的子项返回，其他则忽略。

FileFilter是一个接口，所以当我们需要定义某种过滤规则时，我们可以定义一个类来实现这个接口，而此接口的实例可传递给 File 类的 listFiles(FileFilter) 方法。

* **listFiles方法**

File的listFiles方法用于返回一个抽象路径名数组，这些路径名表示此抽象路径名表示的目录中的文件。其方法定义:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)

File[] listFiles()File[] listFiles()

返回值：抽象路径名数组，这些路径名表示此抽象路径名表示的目录中的文件和目录。如果目录为空，那么数组也将为空。如果抽象路径名不表示一个目录，或者发生 I/O 错误，则返回 null。

\* 设置文件的过滤条件

\* File： 文件

\* Filter: 过滤器

public class Demo{

public static void main(String[] args) {

//FileFilter 是一个接口

//new FileFilter(){} 创建匿名内部类

FileFilter filter = new FileFilter(){

//accept 方法是在FileFilter中定义的抽象方法。

//accept: 接受，测试文件是否被接受

public boolean accept(File file){

//接受一个文件的名是以.txt为结尾的。

String name=file.getName();

return name.endsWith(".txt");

}

};

File file1 = new File("abc/demo1.txt");

File file2 = new File("abc/test.dat");

//检查 file1 使用以 .txt 为结尾

System.out.println(filter.accept(file1));//true

//检查 file2 使用以 .txt 为结尾

System.out.println(filter.accept(file2));//false

//listFiles 重载方法，与filter配合可以过滤文件夹内容列表，列出满足条件的文件

File dir=new File("abc");

//满足过滤条件的全部文件（或文件夹）

File[] files=dir.listFiles(filter);

for(File file:files){

System.out.println(file.getName());

}

//有过滤条件的列目录方法

//listFiles(过滤条件);

12.文件操作——RandomAccessFile

Java提供了一个可以对文件随机访问的操作，访问包括读和写操作。该类名为RandomAccessFile。该类的读写是基于指针的操作。

RandomAccessFile 简称 RAF，Java是提供的API，可以从文件的任意一个位置开始访问文件的内容。

Random： 随机，任意位置 Access： 访问，读写

任何文件都是一个长长的byte数据序列。

* 只读模式

RandomAccessFile在对文件进行随机访问操作时有两个模式，分别为只读模式(只读取文件数据)，和读写模式(对文件数据进行读写)。

在创建RandomAccessFile时，其提供的构造方法要求我们传入访问模式:

[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)

RandomAccessFile(File file,String mode)

RandomAccessFile(String filename,String mode)RandomAccessFile(File file,String mode) RandomAccessFile(String filename,String mode)

其中构造方法的第一个参数是需要访问的文件，而第二个参数则是访问模式:

“r”:字符串”r”表示对该文件的访问是只读的。

* 读写模式

创建一个基于文件访问的读写模式的RandomAccessFile我们只需要在第二个参数中传入”rw”即可。[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)

RandomAccessFile raf = new RandomAccessFile(file,”rw”);RandomAccessFile raf = new RandomAccessFile(file,”rw”);

那么这时在使用RandomAccessFile对该文件的访问就是又可读又可写的。

* **字节数据读写操作**

**1. write(int d)方法**

RandomAccessFile提供了一个可以向文件中写出字节的方法:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)void write(int d)void write(int d)

该方法会根据当前指针所在位置处写入一个字节，是将参数int的”低8位”写出。

**2. read()方法**

RandomAccessFile提供了一个可以从文件中读取字节的方法:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)int read() int read()

该方法会从RandomAccessFile当前指针位置读取一个byte(8位) 填充到int的低八位, 高24位为0, 返回值范围正数: 0~255, 如果返回-1表示读取到了文件末尾EOF(EOF:End Of File)! 每次读取后自动移动文件指针, 准备下次读取。

**3. write(byte[] d)方法**

RandomAccessFile提供了一个可以向文件中写出一组字节的方法:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)void write(byte[] d)void write(byte[] d)

该方法会根据当前指针所在位置处连续写出给定数组中的所有字节，与该方法相似的还有一个常用方法:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)

void write(byte[] d,int offset,int len)void write(byte[] d,int offset,int len)

该方法会根据当前指针所在位置处连续写出给定数组中的部分字节，这个部分是从数组的offset处开始，连续len个字节。

**4. read(byte[] d)方法**

RandomAccessFile提供了一个可以从文件中批量读取字节的方法:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)int read(byte[] b)int read(byte[] b)

该方法会从文件中尝试最多读取给定数组的总长度的字节量，并从给定的字节数组第一个位置开始，将读取到的字节顺序存放至数组中，返回值为实际读取到的字节量 。

**5. close方法**

RandomAccessFile在对文件访问的操作全部结束后，要调用close()方法来释放与其关联的所有系统资源:[copytext](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)pop-upvoid close()

1. void close()

* RandomAccessFile raf = new RandomAccessFile(file,”rw”);
* …..//读写操作
* raf.close();//访问完毕后要关闭以释放系统资源。
* **文件指针操作**

**1. getFilePointer方法**

RandomAccessFile的读写操作都是基于指针的，也就是说总是在指针当前所指向的位置进行读写操作。

RandomAccessFile提供了一个可以获取当前指针位置的方法:[copytext](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)pop-uplong getFilePointer()long getFilePointer()

RandomAccessFile在创建时默认指向文件开始(第一个字节)，通过getFilePointer方法获取指针位置时值是"0"。

例如:

[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)

1. RandomAccessFile raf = new RandomAccessFile(file,”rw”);
2. System.out.println(raf.getFilePointer());//0
3. raf.write(‘A’);//写出一个字节后，指针自动向后移动到下一个字节位置
4. System.out.println(raf.getFilePointer());//1
5. raf.writeInt(3);
6. System.out.println(raf.getFilePointer());//5
7. raf.close();RandomAccessFile raf = new RandomAccessFile(file,”rw”);System.out.println(raf.getFilePointer());//0raf.write(‘A’);//写出一个字节后，指针自动向后移动到下一个字节位置System.out.println(raf.getFilePointer());//1raf.writeInt(3);System.out.println(raf.getFilePointer());//5raf.close();

**2. seek方法**

RandomAccessFile的提供了一个方法用于移动指针位置[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html):void seek(long pos)void seek(long pos)

使用该方法可以移动指针到指定位置。

[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)

1. RandomAccessFile raf = new RandomAccessFile(file,”rw”);
2. System.out.println(raf.getFilePointer());//0
3. raf.write(‘A’);//指针位置1
4. raf.writeInt(3);//指针位置5
5. raf.seek(0);//将指针移动到文件开始处(第一个字节的位置)
6. System.out.println(raf.getFilePointer());//0
7. raf.close(); RandomAccessFile raf = new RandomAccessFile(file,”rw”);System.out.println(raf.getFilePointer());//0raf.write(‘A’);//指针位置1raf.writeInt(3);//指针位置5raf.seek(0);//将指针移动到文件开始处(第一个字节的位置)System.out.println(raf.getFilePointer());//0raf.close();

**3. skipBytes方法**

RandomAccessFile的提供了一个方法可以尝试跳过输入的 n 个字节以丢弃跳过的字节，方法定义为:[copytextpop-up](http://pdf7.tarena.com.cn/tts8_source/ttsPage/JAVA/JSD_N_V06/JAVASE02/DAY01/SUPERDOC/01/index.html)int skipBytes(int n)int skipBytes(int n)

该方法可能跳过一些较少数量的字节（可能包括零）。这可能由任意数量的条件引起；在跳过n个字节之前已到达文件的末尾只是其中的一种可能。此方法不抛出 EOFException。返回跳过的实际字节数。如果 n 为负数，则不跳过任何字节。

* **RAF 写文件案例**

public class Demo {

public static void main(String[] args)

throws Exception{

String file="abc/demo.txt";

//创建RAF对象，以读写方式创建对象时候如果文件不存在，则在磁盘上自动创建文件，文件默认的指针位置在0

//如果文件与文件夹同名或者不能写文件则抛出异常

RandomAccessFile raf=new RandomAccessFile(file, "rw");

//检查文件的读写指针位置

long p=raf.getFilePointer();

System.out.println(p);//0

//将数据写到文件中

raf.write(65); //有效范围：0~255

p = raf.getFilePointer();

System.out.println(p);

raf.write(66);

p = raf.getFilePointer();

System.out.println(p);

long l = raf.length();//检查文件的长度

System.out.println(l);

raf.close();//raf必须关闭

* **RAF 读取文件案例：**

public class Demo{

public static void main(String[] args)

throws Exception {

//以只读访问打开文件

String file = "abc/demo.txt";

RandomAccessFile raf=new RandomAccessFile(file, "r");

//刚刚打开的文件读写指针位置是0

long p=raf.getFilePointer();

System.out.println(p);//0

//读取0位置上的数据(65)

int b = raf.read();//将byte填充到int

//占用int： 0~255 范围

System.out.println(b);//65

//读取以后，文件指针位置自动移动一下

//检查文件指针位置

p = raf.getFilePointer();

System.out.println(p);//1

//读取下一个byte数据: 66

b = raf.read();

System.out.println(b);//

//文件指针位置：

p = raf.getFilePointer();

System.out.println(p);//2

//在文件末尾时候调用raf.read()

//返回一个特殊值：-1 表示读取到文件末尾了

b = raf.read();

System.out.println(b);//-1

p = raf.getFilePointer();

System.out.println(p);//2

raf.close();

}

* RAF 总结
* **理解什么是文件**

文件是由byte组成的序列

RAF可以打开文件，在文件的任意位置开始读写

RAF打开文件方式：

r 只读

rw 读写

* **基本的读写方法**

read() 读取一个byte

write() 写出一个byte

文件的读写位置（文件指针）会自动移动每次第一个byte（0~255）

在基本读写方法之上，扩展了基本类型的读写

readInt writeInt 每次读写4个byte

readLong writeLong 每次读写8个byte

...

* **String 的读写**

内存中的字符串是char数据，不是byte类型

写出字符串：

需要将字符串进行编码（UTF-8）编码为byte数据

然后在写到文件中！！！

读取字符串：

读取byte数据

将byte数据解码（UTF-8）为字符串！

* **随机读写文件**

RAF 可以从文件的任何位置开始读写文件，其核心方法是seek(位置)：

* **基本类型读写**

RAF 提供了基本类型的读写方法，基本类型的读写方法的底层是 read() write() 方法。

int 拆分为 4个byte读写

long 拆分为 8个byte读写

byte为 1个byte读写

short 拆分为 2个byte读写

float 拆分为 4个byte读写

doubel 拆分为 8个byte读写

boolean为 1个byte读写

char 拆分为 2个byte读写

* **int 的读写原理案例：**

\* RAF 整数数据读写，其他类型 略

public class Demo0 {

public static void main(String[] args)

throws Exception{

String file = "abc/integer.dat";

RandomAccessFile raf = new RandomAccessFile(file, "rw");

//将int数据126712 拆分为4个byte写到文件中，文件指针连续移动4次

raf.writeInt(126712);

long p = raf.getFilePointer();

System.out.println(p);

raf.seek(0);

//读取一个整数: 连续读取4个byte，拼接为一个int数据, 文件指针连续移动4次

int n = raf.readInt();

p = raf.getFilePointer();

System.out.println(p);//4

System.out.println(n);//126712

raf.close();

}

}

* **字符串IO**

字符串中存储的是char数据，不能直接IO，需要先进行编码，编码为byte数据在进行读写。常见的编码方案是 UTF-8.

在UTF-8 编码中：英文字符1个byte，中文字符3个byte

文字信息必须经过编码才能写到文件中。读取文件时候需要进行解码处理。 如果编码和解码的规则不一致就会出现乱码问题！！

* **综合案例**

将员工信息写到文件中，并且在读取回来：

public class Demo04 {

public static void main(String[] args) throws IOException{

String file="abc/emp.dat";

RandomAccessFile raf=new RandomAccessFile(file, "rw");

write(raf, 0,"Tom", 10, "男", 100, new Date());

write(raf, 1,"范传奇",30,"男",200,new Date());

raf.close();

}

public static void write(

RandomAccessFile raf, //已经打开的文件

int n, //n = 0 1 2 ... 行号

String name,

int age,

String sex,

int salary,

Date hiredate)throws IOException{

int start=n\*80;//n=0,0 n=1,80 n=2,160

raf.seek(start);//将文件指针移动到每行起始位置

//将name编码，然后写到文件中

byte[] bytes=name.getBytes("UTF-8");

raf.write(bytes); //3 9 10

//写出age

raf.seek(start+32);//跳到age位置

raf.writeInt(age);//写出年龄

bytes = sex.getBytes("UTF-8"); //写出性别

raf.write(bytes);

raf.seek(start+46); //写出薪水

raf.writeInt(salary);

SimpleDateFormat fmt = new SimpleDateFormat("yyyy-MM-dd"); //写出日期

String d = fmt.format(hiredate);

bytes = d.getBytes("UTF-8");

raf.write(bytes);

}

}

public class Demo05 { //读取员工信息文件

public static void main(String[] args) throws Exception {

String file="abc/emp.dat"; //打开文件 emp.dat

RandomAccessFile raf=new RandomAccessFile(file, "r");

Emp e1 = read(raf, 0); //读取第一行 （Tom）

Emp e2 = read(raf, 1); //读取第二行 （范传奇）

System.out.println(e1);

System.out.println(e2);

raf.close();

}

public static Emp read(

RandomAccessFile raf, int n) throws Exception {

int start = n\*80;//读取的起始位置

raf.seek(start);//找到name的起始位置

byte[] bytes = new byte[32];

raf.read(bytes);//读取32个byte

//bytes=[T,o,m,0,0,0,0,0,...0]

String name=new String(bytes,"UTF-8").trim();

int age = raf.readInt();

bytes = new byte[10];

raf.read(bytes);

String sex=new String(bytes,"UTF-8").trim();

int salary = raf.readInt();

bytes = new byte[30];

raf.read(bytes);

String str = new String(bytes, "UTF-8").trim();

SimpleDateFormat fmt = new SimpleDateFormat("yyyy-MM-dd");

Date hiredate=fmt.parse(str);

return new Emp(name,age,sex,salary,hiredate);

}

}

class Emp{

String name;int age;String sex;int salary; Date hiredate;

public Emp(String name, int age, String sex, int salary, Date hiredate) {

this.name = name;

this.age = age;

this.sex = sex;

this.salary = salary;

this.hiredate = hiredate;

}

public String toString() {

return "Emp [name=" + name + ", age=" + age + ", sex=" + sex + ", salary=" + salary + ", hiredate=" + hiredate+ "]";

}

}