java.io.File

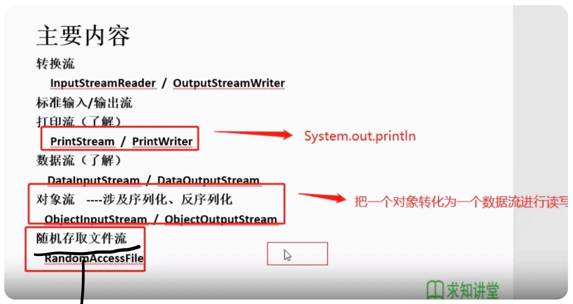
文件流:

FileInputStream / FileOutputStream / FileReader / FileWriter

缓冲流:

BufferedInputStream / BufferedOutputStream /
BufferedReader / BufferedWriter

文件流是基于文件的操作，缓冲流是基于内存的操作



主要内容

转换流

InputStreamReader / OutputStreamWriter

标准输入/输出流

打印流（了解）

PrintStream / PrintWriter → System.out.println

数据流（了解）

DataInputStream / DataOutputStream

对象流 ----涉及序列化、反序列化 → 把一个对象转化为一个数据流进行读写

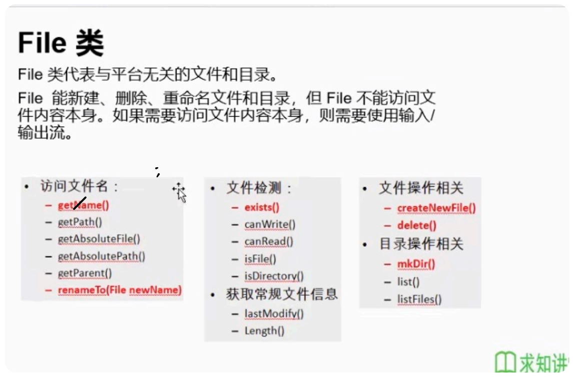
ObjectInputStream / ObjectOutputStream

随机存取文件流

RandomAccessFile

"随机"的意思是可以从文件的任意地方存取

File类用于新建、删除、重命名文件和目录，但不能操作文件内容，



File 类

File 类代表与平台无关的文件和目录。

File 能新建、删除、重命名文件和目录，但 File 不能访问文件内容本身。如果需要访问文件内容本身，则需要使用输入/输出流。

- 访问文件名：
  - getName()
  - getPath()
  - getAbsoluteFile()
  - getAbsolutePath()
  - getParent()
  - renameTo(File newName)

- 文件检测：
  - exists()
  - canWrite()
  - canRead()
  - isFile()
  - isDirectory()
- 获取常规文件信息
  - lastModify()
  - Length()

- 文件操作相关
  - createNewFile()
  - delete()
- 目录操作相关
  - mkDir()
  - list()
  - listFiles()

```
package Test;

import java.io.File;

public class IAndO {
    public static void main(String[] args) {
        File f1 = new File("txt/test0.txt");
        File f2 = new File("txt/");
        System.out.println(f1.getName());
        System.out.println(f2.getName());
        System.out.println(f1.getPath());
        System.out.println(f1.getAbsolutePath());
        System.out.println(f1.getAbsoluteFile());
        System.out.println(f1.getParent());
        System.out.println(f1.getParentFile());
        if(f1.exists()){
            System.out.println("文件"+f1.getName()+"存在！");
        }else {
            System.out.println("访问的文件不存在！");
        }
        if(f1.canWrite()){
            System.out.println("文件"+f1.getName()+"可写！");
        }else {
            System.out.println("访问的文件不可写！");
        }
        if(f1.canRead()){
            System.out.println("文件"+f1.getName()+"可读！");
        }else {
            System.out.println("访问的文件不可读！");
        }
        if(f1.canExecute()){
            System.out.println("文件"+f1.getName()+"可执行！");
        }else {
            System.out.println("访问的文件不可执行！");
        }
        if(f1.exists()){
            System.out.println("文件"+f1.getName()+"存在！");
        }else {
            System.out.println("访问的文件不存在！");
        }
        System.out.println(f1.getName()+"上一次修改时间是"+f1.lastModified());
        f1.renameTo(new File("txt/test1.txt"));

        File f3 = new File("./txt/txt");
        f2.mkdir();  // mkdirs

        String[] fileList1 = f2.list();  // 以字符串数组的形式返回目录下文件/目录名
       File[] fileList2 = f2.listFiles();  // 以File对象返回
    }
}
```

```
package Test;

import java.io.File;

public class BianLi {
    public static void main(String[] args) {
        File directory = new File(".");
        bianLi(directory);
    }
    public static void bianLi(File file){
        if(file.isFile()){
            System.out.println(file.getAbsoluteFile());
        }else{
            System.out.println(file.getAbsoluteFile()+":");
            File[] files = file.listFiles();
            for(File f:files){
                bianLi(f);
            }

        }

    }
}
```

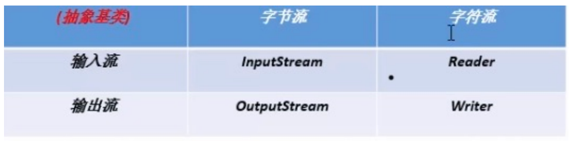
输入输出流:

分类:

根据数据单位分: 字节流(8bit) 字符流(16bit)

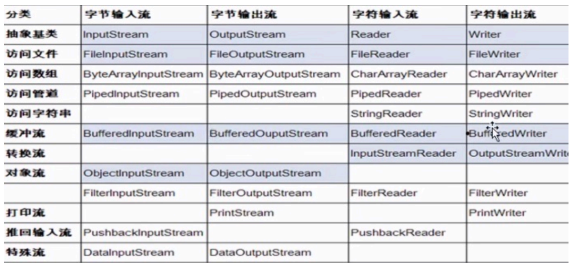
流向分: 输入流、输出流

角色分: 节点流，处理流



(抽象基类)	字节流	字符流
输入流	InputStream	Reader
输出流	OutputStream	Writer

以上4个抽象基类是所有输入输出流的基类



分类	字节输入流	字节输出流	字符输入流	字符输出流
抽象基类	InputStream	OutputStream	Reader	Writer
访问文件	FileInputStream	FileOutputStream	FileReader	FileWriter
访问数组	ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArrayWriter
访问管道	PipedInputStream	PipedOutputStream	PipedReader	PipedWriter
访问字符串			StringReader	StringWriter
缓冲流	BufferedInputStream	BufferedOuputStream	BufferedReader	BufferedWriter
转换流			InputStreamReader	OutputStreamWrit
对象流	ObjectInputStream	ObjectOutputStream		
	FilterInputStream	FilterOutputStream	FilterReader	FilterWriter
打印流		PrintStream		PrintWriter
推回输入流	PushbackInputStream		PushbackReader	
特殊流	DataInputStream	DataOutputStream		

字节流的输入输出:

```
package Test;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.nio.charset.StandardCharsets;

public class IAndOStream {
    public static void main(String[] args) {
        testFileInputStream();
        testFileOutputStream();
    }
    public static void testFileInputStream(){
        try{
            FileInputStream in = new FileInputStream("txt/test1.txt");
            int len;
            byte[] b = new byte[10];
            while((len = in.read(b))!=-1){    // 第二次读取到 len = -1
                // 将byte转为string
                System.out.println(new String(b, 0, len));
            }
            in.close();
        }catch (Exception e){
            e.printStackTrace();
        }

    }
    public static void testFileOutputStream(){
        try{
            FileOutputStream out = new FileOutputStream("txt/test2.txt");
            String str = new String("Hello");
            out.write(str.getBytes(StandardCharsets.UTF_8));
            out.flush();
            out.close();
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

字符流的输入与输出：

字符流与字节流的区别在于内存中的中间载体由byte变为了char.

```
package Test;

import java.io.FileReader;
import java.io.FileWriter;
import java.nio.charset.StandardCharsets;

public class IAndOStream2 {
    public static void main(String[] args) {
        testFileInputStream();
        testFileOutputStream();

    }
    public static void testFileInputStream(){
        try{
            FileReader in = new FileReader("txt/test1.txt");
            int len;
            char[] b = new char[10];
            while((len = in.read(b))!=-1){
                // 将byte转为string
                System.out.println(new String(b, 0, len));
            }
            in.close();
        }catch (Exception e){
            e.printStackTrace();
        }

    }
    public static void testFileOutputStream(){
        try{
            FileWriter out = new FileWriter("txt/test3.txt");
            String str = new String("Hello");
            out.write(str);
            out.flush();
            out.close();
        }catch (Exception e){
            e.printStackTrace();
        }
    }
}.
```

缓冲流：

上述的字节流和字符流都是CPU直接与外存交互，速度极慢。

而缓冲流则是在内存中建立一个缓冲区，CPU与内存交互，速度较快

BufferedInputStream	BufferedOutputStream
BufferedReader	BufferedWriter

例：FileInputStream in = new FileInputStream("____");

BufferedInputStream br = new BufferedInputStream(in);

随后的操作类似

关闭流时遵循先进后出的顺序.

转换流：

转换流提供了在字节流和字符流之间的转换

InputStreamReader 和 OutputStreamWriter

例:

```
FileInputStream fs = new FileInputStream( );
InputStreamReader in = new InputStreamReader(fs, "文件编码");
        . . . . . .
    in.close;
    fs.close;
```

标准输入流和输出流:

```
InputStreamReader is = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(is);
String str = "";
while((str= br.readLine())!=null){
        System.out.println(str);
}
 br.close;
is.close;
}
```

数据流:

```
import java.io.*;

public class DataOut {
   public static void main(String[] args) {
      try{
         testDataOutputStream();
      } catch (IOException e) {
         e.printStackTrace();
      }
      try{
         testDataInputStream();
      } catch (IOException e) {
         e.printStackTrace();
      }
```

```
    }

    public static void testDataOutputStream() throws IOException {
        DataOutputStream out = new DataOutputStream(new FileOutputStream("./txt/data.txt"));
        out.writeBoolean(true);
        out.flush();
        out.close();
    }
    public static void testDataInputStream() throws IOException {
        DataInputStream in = new DataInputStream(new FileInputStream("./txt/data.txt"));
        System.out.println(in.readBoolean());
        in.close();
    }
}
```

对象流：

涉及到序列化和反序列化

Serialize 用ObjectOutputStream将一个Java对象写入IO流

Deserialize 用ObjectInputStream 将序列化对象从IO中恢复

不能序列化static 和transient (static是属于类的，而不属于对象)

反序列化和序列化的类要严格一致：类名，包名，机构名.

```
import java.io.*;

public class ObjectStream {
    public static void main(String[] args) {
        try {
            testSerialize();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            testDeSerialize();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
    public static void testSerialize() throws IOException {
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("./txt/serialize.txt"));
        Person person = new Person();
        person.name="Ida";
        person.age = 21;
        out.writeObject(person);
        out.flush();
        out.close();
    }
    public static void testDeSerialize() throws IOException, ClassNotFoundException {
        ObjectInputStream in = new ObjectInputStream(new FileInputStream("./txt/serialize.txt"));
        Person person = (Person)in.readObject();
        System.out.println(person.name);
        System.out.println(person.age);
    }
}
```

随机存取流：

RondamAccessFile

支持文件的任意地方读、写文件.

构造器：

```
public RandomAccessFile(File file, String mode);
public RandomAccessFile(String name, String mode);
```

r } 不用flush
rw
rwd
rws