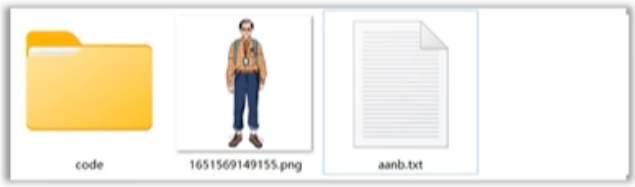
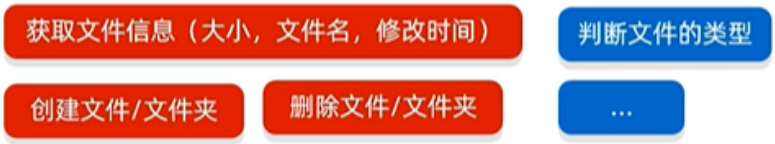
I/O流

存储数据的方案

File

- File是java.io.包下的类， File类的对象，用于代表当前操作系统的文件（可以是文件、或文件夹）。

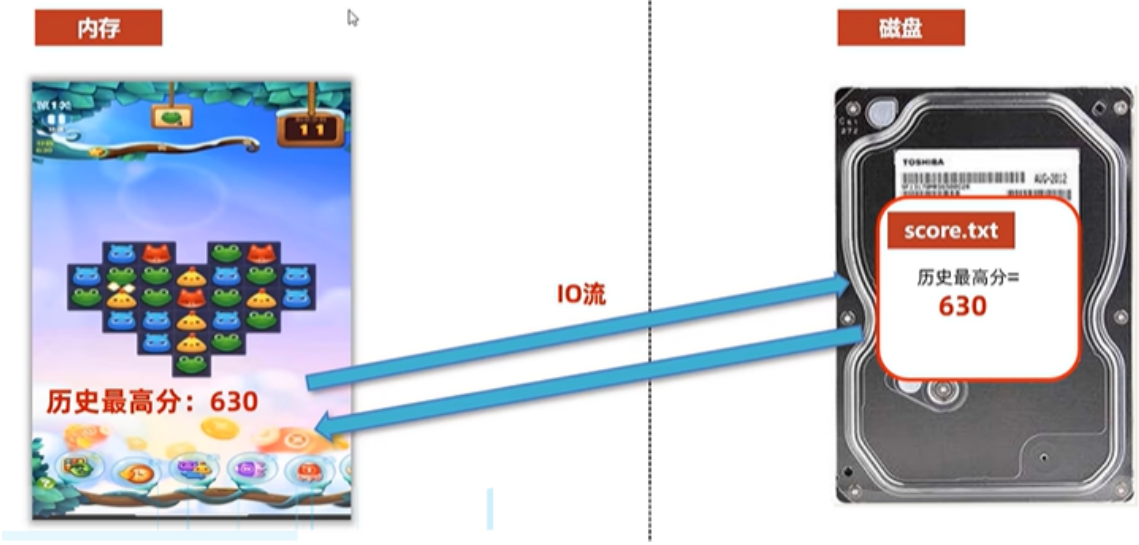




注意：File类只能对文件本身进行操作，不能读写文件里面存储的数据。

IO流

- 用于读写数据的（可以读写文件，或网络中的数据...）



File：代表文本

IO流：读写数据

```java
public class FileTest1 {
    public static void main(String[] args) {
        // 1、创建一个File对象，指代某个具体的文件。
        // 路径分隔符
        // File f1 = new File("D:/resource/ab.txt");
        // File f1 = new File("D:\\resource\\ab.txt");
        File f1 = new File( pathname: "D:" + File.separator +"resource" + File.separator + "ab.txt");
        System.out.println(f1.length()); // 文件大小

        File f2 = new File( pathname: "D:/resource");
        System.out.println(f2.length());

        // 注意：File对象可以指代一个不存在的文件路径
        File f3 = new File( pathname: "D:/resource/aaaa.txt");
        System.out.println(f3.length());
        System.out.println(f3.exists()); // false

        // 我现在要定位的文件是在模块中，应该怎么定位呢？
        // 绝对路径：带盘符的
        // File f4 = new File("D:\\code\\javasepromax\\file-io-app\\src\\itheima.txt");
        // 相对路径（重点）：不带盘符，默认是直接去工程下寻找文件的。
        File f4 = new File( pathname: "file-io-app\\src\\itheima.txt");
        System.out.println(f4.length());
    }
}
```

创建File类的对象

构造器	说明
public File(String pathname)	根据文件路径创建文件对象
public File(String parent, String child)	根据父路径和子路径名字创建文件对象
public File(File parent, String child)	根据父路径对应文件对象和子路径名字创建文件对象

注意

- File对象既可以代表文件、也可以代表文件夹。
- File封装的对象仅仅是一个路径名，这个路径可以是存在的，也允许是不存在的。

绝对路径、相对路径

- 绝对路径：从盘符开始

```java
File file1 = new File("D:\\itheima\\a.txt");
```

- 相对路径：不带盘符，默认直接到当前工程下的目录寻找文件。

```java
File file3 = new File("模块名\\a.txt");
```

一、常见方法1：判断文件类型、获取文件信息

```
public class FileTest2 {
    public static void main(String[] args) throws UnsupportedEncodingException {
        // 1.创建文件对象，指代某个文件
        // File f1 = new File("D:/resource/ab.txt");
        File f1 = new File( pathname: "D:/resource/");

        // 2、public boolean exists()：判断当前文件对象，对应的文件路径是否存在，存在返回true.
        System.out.println(f1.exists());

        // 3、public boolean isFile() ：判断当前文件对象指代的是否是文件，是文件返回true，反之.
        System.out.println(f1.isFile());

        // 4、public boolean isDirectory()  ：判断当前文件对象指代的是否是文件夹，是文件夹返回true，反之.
        System.out.println(f1.isDirectory());

        // 5.public String getName()：获取文件的名称（包含后缀）
        System.out.println(f1.getName());

        // 6.public long length()：获取文件的大小，返回字节个数
        System.out.println(f1.length());

        // 7.public long lastModified()：获取文件的最后修改时间。
        long time = f1.lastModified();
        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy/MM/dd HH:mm:ss");
        System.out.println(sdf.format(time));

        // 8.public String getPath()：获取创建文件对象时，使用的路径
        File f2 = new File( pathname: "D:\\resource\\ab.txt");
        File f3 = new File( pathname: "file-io-app\\src\\itheima.txt");
        System.out.println(f2.getPath());
        System.out.println(f3.getPath());

        // 9.public String getAbsolutePath()：获取绝对路径
        System.out.println(f2.getAbsolutePath());
        System.out.println(f3.getAbsolutePath());
```

```
true
true
false
ab.txt
71
2022/04/13 23:16:23
D:\resource\ab.txt
file-io-app\src\itheima.txt
D:\resource\ab.txt
D:\code\javasepromax\file-io-app\src\itheima.txt
```

方法名称	说明
public boolean exists()	判断当前文件对象，对应的文件路径是否存在，存在返回true
public boolean isFile()	判断当前文件对象指代的是否是文件，是文件返回true，反之。
public boolean isDirectory()	判断当前文件对象指代的是否是文件夹，是文件夹返回true，反之。
public String getName()	获取文件的名称（包含后缀）
public long length()	获取文件的大小，返回字节个数
public long lastModified()	获取文件的最后修改时间。
public String getPath()	获取创建文件对象时，使用的路径
public String getAbsolutePath()	获取绝对路径

二、常见方法2：创建文件、删除文件

```
File f1 = new File( pathname: "D:/resource/itheima2.txt");
System.out.println(f1.createNewFile());
```

//warning:担心这个路径是乱写的，所以抛异常提醒

```
// 1、public boolean createNewFile()：创建一个新文件（文件内容为空），创建成功返回true,反之。
File f1 = new File( pathname: "D:/resource/itheima2.txt");
System.out.println(f1.createNewFile());
```

//warnjing:如果该文件存在，则创建失败，返回false

```
public class FileTest3 {
    public static void main(String[] args) throws Exception {
        // 1、public boolean createNewFile()：创建一个新文件（文件内容为空），创建成功返回true,反之。
        File f1 = new File( pathname: "D:/resource/itheima2.txt");
        System.out.println(f1.createNewFile());

        // 2、public boolean mkdir()：用于创建文件夹，注意：只能创建一级文件夹
        File f2 = new File( pathname: "D:/resource/aaa");
        System.out.println(f2.mkdir());

        // 3、public boolean mkdirs()：用于创建文件夹，注意：可以创建多级文件夹
        File f3 = new File( pathname: "D:/resource/bbb/ccc/ddd/eee/fff/ggg");
        System.out.println(f3.mkdirs());

        // 3、public boolean delete()：删除文件，或者空文件，注意：不能删除非空文件夹。
        System.out.println(f1.delete());
        System.out.println(f2.delete());
        File f4 = new File( pathname: "D:/resource");
        System.out.println(f4.delete());
    }
}
```

putout:

```
true
false
false
true
true
```

File类创建文件的功能

方法名称	说明
public boolean createNewFile()	创建一个新的空的文件
public boolean mkdir()	只能创建一级文件夹
public boolean mkdirs()	可以创建多级文件夹

File类删除文件的功能

方法名称	说明
public boolean delete()	删除文件、空文件夹

注意：delete方法默认只能删除文件和空文件夹，删除后的文件不会进入回收站。

三、遍历文件夹

```java
public class FileTest4 {
    public static void main(String[] args) {
        // 1、public String[] list()：获取当前目录下所有的"一级文件名称"到一个字符串数组中去返回。
        File f1 = new File( pathname: "D:\\course\\待研发内容");
        String[] names = f1.list();
        for (String name : names) {
            System.out.println(name);
        }

        // 2、public File[] listFiles()：（重点）获取当前目录下所有的"一级文件对象"到一个文件对象数组中去返回（重点）
        File[] files = f1.listFiles();
        for (File file : files) {
            System.out.println(file.getAbsolutePath());
        }
    }
}
```

File类提供的遍历文件夹的功能

方法名称	说明
public String[] list()	获取当前目录下所有的"一级文件名称"到一个字符串数组中去返回。
public File[] listFiles()	获取当前目录下所有的"一级文件对象"到一个文件对象数组中去返回（重点）

使用listFiles方法时的注意事项：

- 当主调是文件，或者路径不存在时，返回null
- 当主调是空文件夹时，返回一个长度为0的数组
- 当主调是一个有内容的文件夹时，将里面所有一级文件和文件夹的路径放在File数组中返回
- 当主调是一个文件夹，且里面有隐藏文件时，将里面所有文件和文件夹的路径放在File数组中返回，包含隐藏文件
- 当主调是一个文件夹，但是没有权限访问该文件夹时，返回null

四、字符集

标准ASCII字符集

- ASCII(American Standard Code for Information Interchange)：美国信息交换标准代码，包括了英文、符号等。
- 标准ASCII使用1个字节存储一个字符，首尾是0，总共可表示128个字符，对美国佬来说完全够用。

中国人 还不够我塞牙缝！

GBK（汉字内码扩展规范，国标）

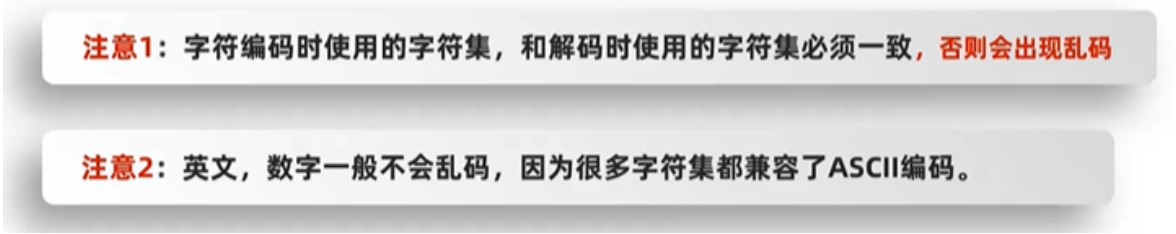
- 汉字编码字符集，包含了2万多个汉字等字符，GBK中一个中文字符编码成两个字节的形式存储。
- 注意：GBK兼容了ASCII字符集。

Unicode字符集(统一码，也叫万国码)

- Unicode是国际组织制定的，可以容纳世界上所有文字、符号的字符集。

UTF-32 我4个字节表示一个字符 有容乃大

奢侈！ 占存储空间，通信效率变低！



1、字符集的解码、操作

Java代码完成对字符的编码

String提供了如下方法	说明
byte[] getBytes()	使用平台的默认字符集将该 String编码为一系列字节，将结果存储到新的字节数组中
byte[] getBytes(String charsetName)	使用指定的字符集将该 String编码为一系列字节，将结果存储到新的字节数组中

Java代码完成对字符的解码

String提供了如下方法	说明
String(byte[] bytes)	通过使用平台的默认字符集解码指定的字节数组来构造新的 String
String(byte[] bytes, String charsetName)	通过指定的字符集解码指定的字节数组来构造新的 String

```
// 按照指定字符集进行编码。
byte[] bytes = data.getBytes( charsetName: "GBK");
```

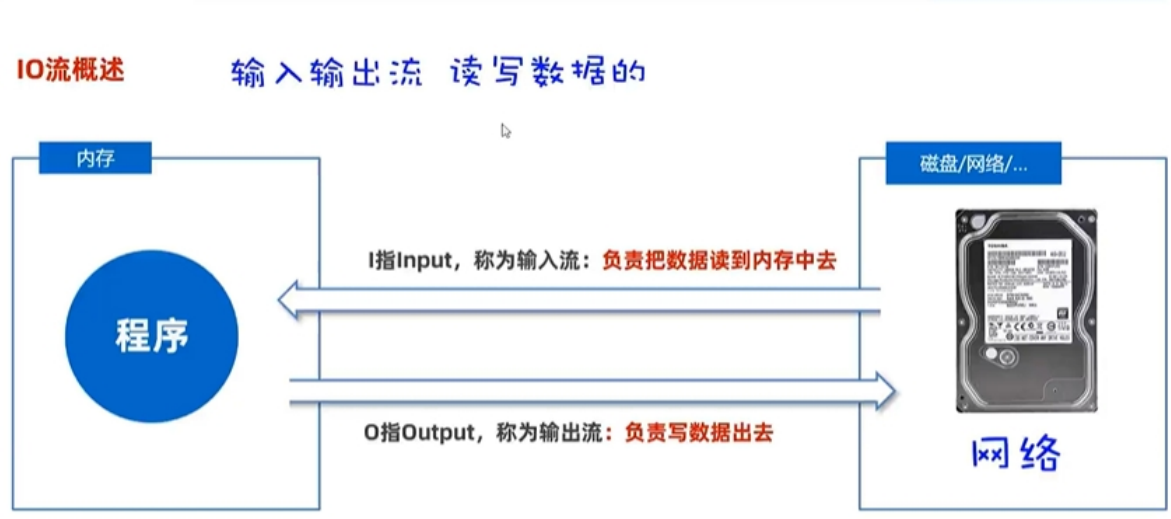
//warning:担心把编码名称写错，所以报错

```java
public class Test {
    public static void main(String[] args) throws Exception {
        // 1、编码
        String data = "a我b";
        byte[] bytes = data.getBytes(); // 默认是按照平台字符集（UTF-8）进行编码的。
        System.out.println(Arrays.toString(bytes));

        // 按照指定字符集进行编码。
        byte[] bytes1 = data.getBytes( charsetName: "GBK");
        System.out.println(Arrays.toString(bytes1));

        // 2、解码
        String s1 = new String(bytes); // 按照平台默认编码（UTF-8）解码
        System.out.println(s1);

        String s2 = new String(bytes1);
        System.out.println(s2);

    }
}
```

putout:

```
[97, -26, -120, -111, 98]
[97, -50, -46, 98]
a我b
a��b
```

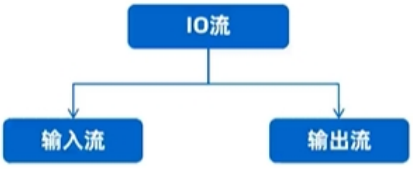
五、I/O流



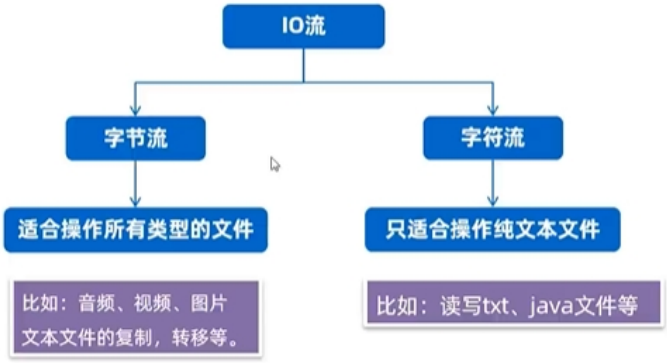
IO流的应用场景

IO流的分类

按流的方向分为：



按流中数据的最小单位，分为：

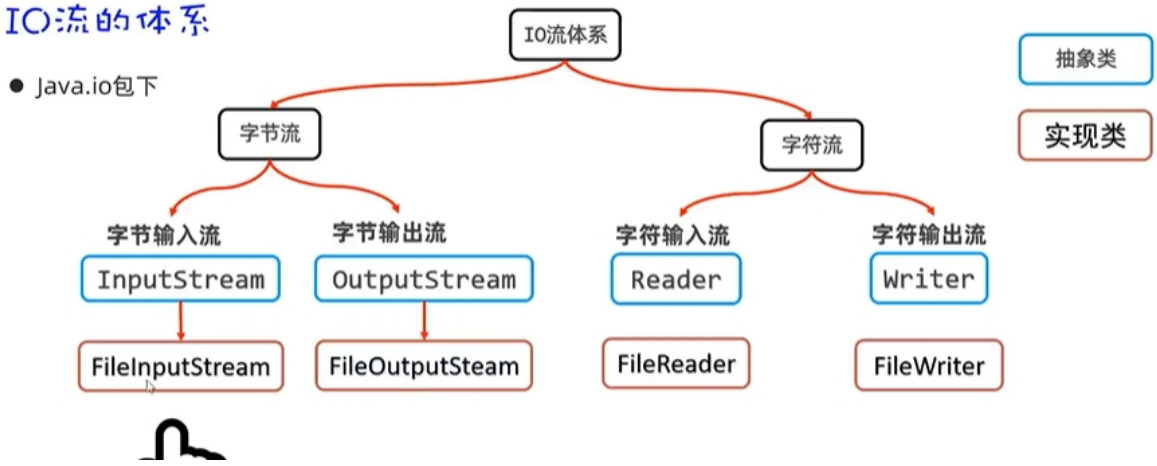


IO流总体来看就有四大流

字节输入流　字节输出流　字符输入流　字符输出流

总结流的四大类：

- 字节输入流：以内存为基准，来自磁盘文件/网络中的数据以字节的形式读入到内存中去的流
- 字节输出流：以内存为基准，把内存中的数据以字节写出到磁盘文件或者网络中去的流。
- 字符输入流：以内存为基准，来自磁盘文件/网络中的数据以字符的形式读入到内存中去的流。
- 字符输出流：以内存为基准，把内存中的数据以字符写出到磁盘文件或者网络介质中去的流。



1、字节流

FileInputStream

FileInputStream（文件字节输入流）

- 作用：以内存为基准，可以把磁盘文件中的数据以字节的形式读入到内存中去。

FileInputStream（文件字节输入流）

- 作用：以内存为基准，可以把磁盘文件中的数据以字节的形式读入到内存中去。

构造器	说明
public FileInputStream(File file)	创建字节输入流管道与源文件接通
public FileInputStream(String pathname)	创建字节输入流管道与源文件接通

方法名称	说明
public int read()	每次读取一个字节返回，如果发现没有数据可读会返回-1.
public int read(byte[] buffer)	每次用一个字节数组去读取数据，返回字节数组读取了多少个字节，如果发现没有数据可读会返回-1.

```
public class FileInputStreamTest1 {
    public static void main(String[] args) throws Exception {
        // 1、创建文件字节输入流管道，与源文件接通。
        // InputStream is = new FileInputStream(new File("file-io-app\\src\\itheima01.txt"));
        // 简化写法：推荐使用。
        InputStream is = new FileInputStream(("file-io-app\\src\\itheima01.txt"));

        // 2、开始读取文件的字节数据。
        // public int read():每次读取一个字节返回，如果没有数据了，返回-1.
        int b1 = is.read();
        System.out.println((char)b1);

        int b2 = is.read();
        System.out.println((char) b2);

        int b3 = is.read();
        System.out.println(b3);
    }
}
```

```
// 3、使用循环改造上述代码
int b; // 用于记住读取的字节。
while ((b = is.read()) != -1){
    System.out.print((char) b);
}

// 读取数据的性能很差！
// 读取汉字输出会乱码！！无法避免的！！
// 流使用完毕之后，必须关闭！释放系统资源！
is.close();
```

```java
// 2、开始读取文件中的字节数据：每次读取多个字节。
byte[] buffer = new byte[3];
int len = is.read(buffer);
String rs = new String(buffer);
System.out.println(rs);
System.out.println("当次读取的字节数量：" + len);

// buffer = [abc]
// buffer = [66c]
int len2 = is.read(buffer);
// 注意：读取多少，倒出多少。
String rs2 = new String(buffer, offset: 0, len2);
System.out.println(rs2);
System.out.println("当次读取的字节数量：" + len2);

int len3 = is.read(buffer);
System.out.println(len3);
```

//warning:rs2那里如果没有加0，len2就会输出66c因为byte存储数据的结构为队列

//warning:这里的len2指的是读取buffer的字节

```java
    // 3、使用循环改造。
    byte[] buffer = new byte[3];
    int len; // 记住每次读取了多少个字节。  abc 66
    while ((len = is.read(buffer)) != -1){
        // 注意：读取多少，倒出多少。
        String rs = new String(buffer, offset: 0 , len);
        System.out.print(rs);
    }
    // 性能得到了明显的提升！！
    // 这种方案也不能避免读取汉字输出乱码的问题！！

    is.close(); // 关闭流
}
}
```

想不乱码，解决方案一：定义一个与文件一样大的字节数组，一次性读取完文件的全部字节。

```java
public class FileInputStreamTest3 {
    public static void main(String[] args) throws Exception {
        // 1、一次性读取完文件的全部字节到一个字节数组中去。
        // 创建一个字节输入流管道与源文件接通
        InputStream is = new FileInputStream( name: "file-io-app\\src\\itheima03.txt");

        // 2、准备一个字节数组，大小与文件的大小正好一样大。
        File f = new File( pathname: "file-io-app\\src\\itheima03.txt");
        long size = f.length();
        byte[] buffer = new byte[(int) size];

        int len = is.read(buffer);
        System.out.println(new String(buffer));

        System.out.println(size);
        System.out.println(len);
    }
}
```

文件字节输入流：一次读取完全部字节

- **方式一：**自己定义一个字节数组与被读取的文件大小一样大，然后使用该字节数组，一次读完文件的全部字节。

方法名称	说明
public int read(byte[] buffer)	每次用一个字节数组去读取，返回字节数组读取了多少个字节，如果发现没有数据可读会返回-1.

- **方式二：**Java官方为InputStream提供了如下方法，可以直接把文件的全部字节读取到一个字节数组中返回。

方法名称	说明
public byte[] readAllBytes() throws IOException	直接将当前字节输入流对应的文件对象的字节数据装到一个字节数组返回

```java
byte[] buffer = is.readAllBytes();
System.out.println(new String(buffer));
```

- **如果文件过大，创建的字节数组也会过大，可能引起内存溢出。**

读写文本内容更适合用 字符流

字节流 适合做数据的转移，如：文件复制等。

FileOutputStream

FileOutputStream(文件字节输出流)

- 作用：以内存为基准，把内存中的数据以字节的形式写出到文件中去。