1. 对象数组
   1. 对象数组就是数组里的每个元素都是类的对象，赋值时先定义对象，然后将对象直接赋给数组就行了。
   2. 实现：

public class Student

{

// 成员变量

private String name;

private int age;

// 构造方法

public Student(){}

public Student(String name, int age)

{

this.name = name;

this.age = age;

}

// 成员方法

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public int getAge()

{

return age;

}

public void setAge(int age)

{

this.age = age;

}

@Override

public String toString()

{

return "Student [name=" + name + ", age=" + age + "]";

}

}

/\*\*

把5个学生的信息存储到数组中，并遍历数组，获取得到每一个学生信息。

\* 学生:Student

\* 成员变量:name,age

\* 构造方法:无参,带参

\* 成员方法:getXxx()/setXxx()

\* 分析:

\* A:创建学生类。

\* B:创建学生数组(对象数组)。

\* C:创建5个学生对象，并赋值。

\* D:把C步骤的元素，放到数组中。

\* E:遍历学生数组。

\* \*/

public class Practice

{

public static void main(String[] args)

{

// 创建学生数组(对象数组)。

Student[] students = new Student[5];

// 创建5个学生对象，并赋值。

Student s1 = new Student("小明", 27);

Student s2 = new Student("小红", 30);

Student s3 = new Student("小强", 30);

Student s4 = new Student("旺财", 12);

Student s5 = new Student("张三", 35);

// 将对象放到数组中。

students[0] = s1;

students[1] = s2;

students[2] = s3;

students[3] = s4;

students[4] = s5;

// 遍历

for (int x = 0; x < students.length; x++)

{

//System.out.println(students[x]);

Student s = students[x];

System.out.println(s.getName()+"---"+s.getAge());

}

}

}

1. String详解
   1. 定义：String即是字符串，是一连串的字符序列。
   2. 特点：这是一个不可变类，String对象创建后，包含在这个对象内的字符序列不可以改变
   3. 方法：
      1. String concat（String str）

String s1=”hello”;String s2=”world”;

System.out.println(s1.concat(s2));

* + 1. Boolean endsWith/startsWith(String suffix)

String s1=”Quanta”;

System.out.println(s1.startsWith(“Q”)/s1.endsWith(“a”));

* + 1. Boolean equals(String astring)

String s1=”Hi”;String s2=”Hello”;

System.out.println(s1.equals(s2);

* + 1. Int indexOf()/int lastIndexOf()

String s1=”Quanta”;

System.out.println(s1.indexOf(“a”)/s1.lastIndexOf(“a”));

* + 1. String replace(char oldChar,char newChar)

String s2=”Happy”;

System.out.println(s2.replace(“p”,””);

* + 1. String substring(int index)

String s1=”Study”;

System.out.println(s1.substring(3));

* + 1. Char[] toCharArray()

String s1=”GDUFS”;

Char[] convert=s1.toCharArray();

* + 1. String toLowerCase()/toUpperCase()

String s1=”Woooooo”;

System.out.println(s1.toUpperCase());

1. IO流
   1. File类
      1. 文件名相关方法
         1. String getName()
         2. String getPath()
         3. String getAbsoluteFile()
         4. String getParent()
         5. \*Boolean renameTo(File newName)
      2. 文件检测相关方法
         1. Boolean exists()
         2. Boolean canWrite()
         3. Boolean canRead()
         4. Boolean isFIle
         5. Boolean isDirectory
      3. 文件操作
         1. Boolean createNewFile()
         2. Boolean delete()
      4. 目录操作相关的方法
         1. Boolean mkdir()
         2. String[] list()
         3. File [] listFiles()
   2. Io流
      1. File类

**File(String pathname)**

参数：pathname - 路径名字符串

* + 1. 字节流和字符流

前者由InputStream和OutputStream作为基类，后者主要由Reader和Writer作为基类

* + 1. 输入流和输出流
       1. FileInputStream

int read()读取一个字节，如已到文件末尾，则返回 -1

int read(byte[] b)读取多个字节到b数组中，若末尾返回-1

int read(byte[] b,int off,int len) 读取多个字节到b数组中，若末尾返回-1,off - 数据的起始偏移量。len - 读取的最大字节数。

void close() 关闭此文件输入流并释放所有系统资源

* + - 1. FileOutputStream

void write(int b) 将指定字节写入此文件输出流

void write(byte[] b)指定字节数组写入此文件输出流中

void flush() 刷新此输出流并强制写出所有缓冲的输出字节

void close() 关闭此文件输入流并释放所有系统资源

* + - 1. FileReader

int read() 读取单个字符，如已到文件末尾，则返回 -1

int read(char[] cbuf)

读取多个字符到chuf数组中，若末尾返回-1

int read(char[] cbuf,int offset,int length)

读取多个字符到chuf数组中，若末尾返回-1

offset - 以其处开始存储字符的偏移量

length - 要读取的最大字符数

String getEncoding() 此流使用的字符编码的名称

void close() 关闭此文件输入流并释放所有系统资源

4.FileWriter

void write(int c 将指定字符写入此文件输出流

void write(String str,int off,int len)

指定字符串写入此文件输出流中

off - 数据的起始偏移量。

len - 读取的最大字节数。

String getEncoding() 此流使用的字符编码的名称

void flush() 刷新此输出流并强制写出所有缓冲的输出字节

void close() 关闭此文件输入流并释放所有系统资源

1. **class** IODemo
2. {
3. **public** **static** **void** main(String[] args)
4. {
5. **try**
6. {
7. //使用FileInputStream和FileOutputStream进行文件复制
8. FileInputStream fis=**new** FileInputStream("a.txt");
9. FileOutputStream fos=**new** FileOutputStream("b.txt");
10. **int** read;
11. //read=fis.read();
12. **byte** b[]=**new** **byte**[1024];
13. //读取文件，存入字节数组b，返回读取到的字符数，存入read,默认每次将b数组装满
14. read=fis.read(b);
15. **while**(read!=-1)
16. {
17. fos.write(b,0,read);
18. read=fis.read(b);
19. //read=fis.read();
20. }
21. fis.close();
22. fos.close();
23. }
24. **catch** (IOException e)
25. {
26. e.printStackTrace();
27. }
29. }
30. }
    * 1. 处理流与节点流

节点流直接连接到实际的数据源，处理流是对一个已经存在的流进行封装后再实现功能

* + 1. 转换流(字节流转换为字符流)

InputStreamReader(InputStream in);

OutputStreamWriter(OutputStream out);