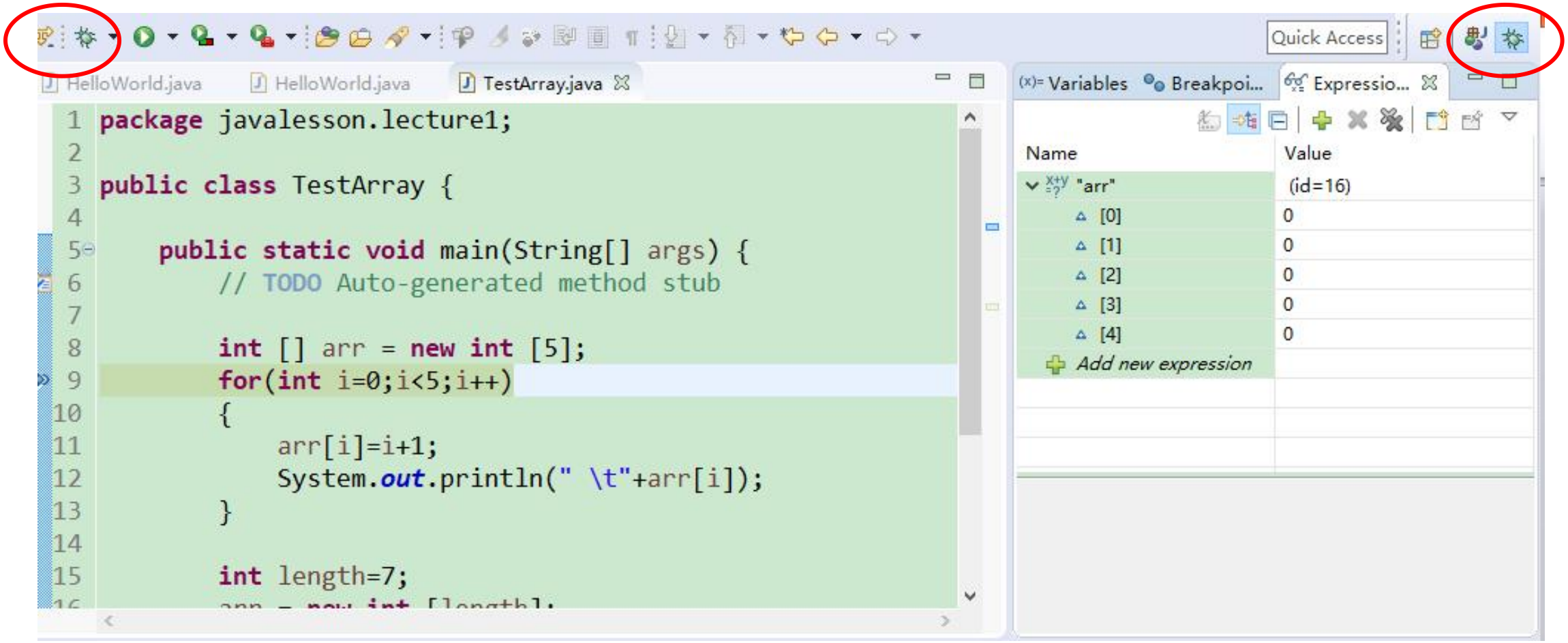
Java语法基础

本节概要

- 使用例子
- 数组与字符串
- Java文件IO
- Java容器变量

Debug调试


Quick Access
HelloWorld.java
HelloWorld.java
TestArray.java
Variables
Breakpoi...
Expressio...
Name
Value
"arr"
(id=16)
[0]
0
[1]
0
[2]
0
[3]
0
[4]
0
Add new expression
1 package javalesson.lecture1;
2
3 public class TestArray {
4
5 public static void main(String[] args) {
6 // TODO Auto-generated method stub
7
8 int [] arr = new int [5];
9 for(int i=0;i<5;i++)
10 {
11 arr[i]=i+1;
12 System.out.println(" \t"+arr[i]);
13 }
14
15 int length=7;

细节

- 经常保存(ctrl+s)，不要轻易删除文件（或代码）。
- & 与 && 的区别。
- 运算符先后顺序。
- 用 equal来判断String是否相等，可类比于 c 的 == 和 strcmp的区别

清華大學 Tsinghua University

数组与字符串

数组基础

- “数组”(Array)是程序语言的一种基本数据结构，它是一种有序组织的数据结构。
- Java数组是Array对象，属于引用类型，也即Java数组变量值是一个指向数组的地址的引用。

1. Java 数组与C语言数组类似，下标 从0开始
2. Java 数组的元素既可以是基本数据类型，也可以是复杂对象　　String [] strs = new String[5];

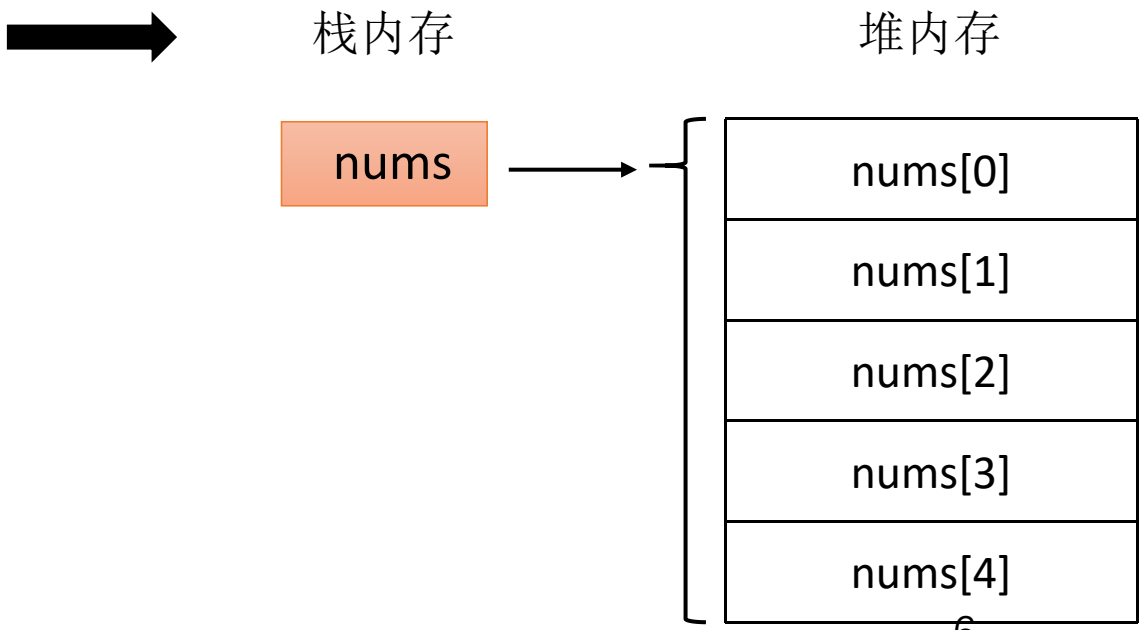
c/c++ 的写法

int nums[5]

int * num = int [5];

delete []num

int[] nums = new int[5];



数组基础

◆高维数组的定义

二维数组其实就是一个特殊的一维数组，一维数组中每个元素就是一个一维数组

```
int [][] num = new int[100][2];

for(int i = 0;i < num.length;i++){
for(int j = 0;j < num[i].length;j++){
        System.out.println(i + " " + j + " " + num[i][j]);
        }
}
```

```
int [][][] num = new int[100][2][4];

for(int i = 0;i < num.length;i++){
    for(int j = 0;j < num[i].length;j++){
        System.out.println(i + " " + j + " " + num[i][j][1][1]);
    }
}
```

字符串类（可参考 const char *）

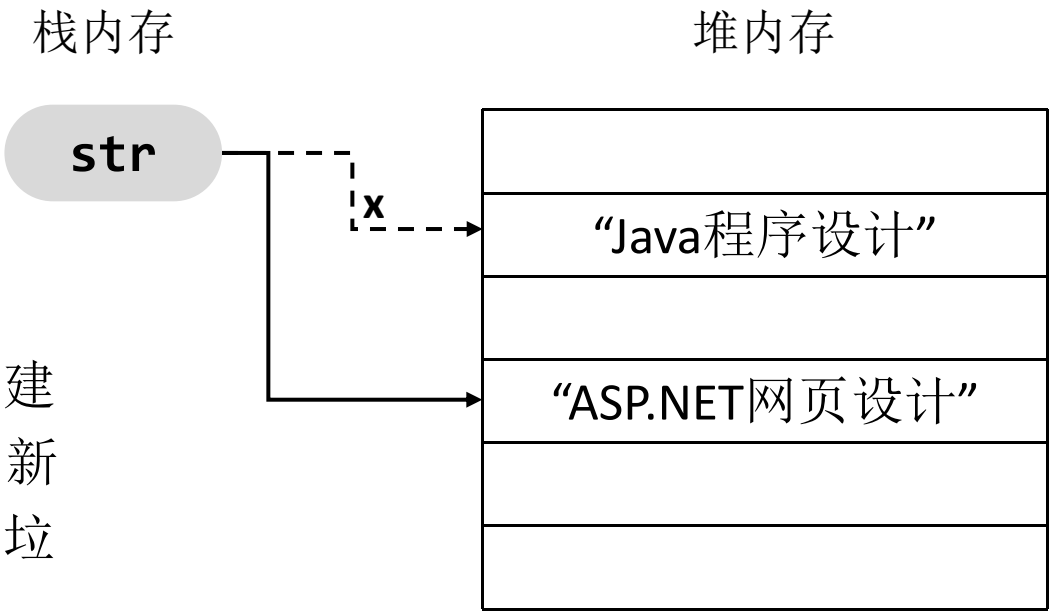
◆ Java 字符串是一个 String 对象

如何理解这句话？

Java 字符串属于一种引用数据类型，所以一旦创建字符串后，就无法改变其值

■ 例子：

```
String str = “JAVA 程序设计”;
str = “ASP.NET网页设计”;
```



字符串的值不是已经改变了吗？

实际上，字符串的值并没有改变，而是新建了另一个字符串，将str这一引用改为指向新的字符串对象。老的字符串对象会被自动垃圾回收。

String 对象构造

◆ String 类提供了多种形式的构造函数

构造函数	说明
String()	创建空字符串
String(String)	使用由”“括起来的字符串或者其他字符串对象 String 来创建字符串对象
String (char[])	使用字符数组创建字符串对象
String (byte[])	使用字节数组创建字符串对象

String 对象处理

◆字符串长度与大小写转换

方法	说明
int length()	取得字符串长度
String toLowerCase()	将字符串的英文字母转换成小写
String toUpperCase()	将字符串的英文字母转换成大写

◆例子

```
String str1 = new String("程序语言的程序设计");
// 显示字符串长度和大小写转换
System.out.print("str1长度:"+str1.length());
System.out.print("转小写:"+str1.toLowerCase());
System.out.println("/转大写:"+str1.toUpperCase());
```

String 对象处理

◆ 子字符串与字符查找

String 对象提供多种查找方法，可以在字符串中找寻所需的字符或者子字符串。

方法	说明
int indexOf(char)	返回第1次查找到字符的索引位置，不存在则返回-1
int lastIndexOf(char)	返回从后往前第1次查找到字符的索引位置，不存在则返回-1
int indexOf(char, int)	返回第1次查找到字符的索引位置，不存在则返回-1；传入参数char为目标字符，int为开始查找的位置
int lastIndexOf(char, int)	返回从后往前第1次查找到字符的索引位置，不存在则返回-1；传入参数char为目标字符，int为开始查找的位置

类似地，String还有四个重载的方法对应以上四个方法，传入参数中的char替换为String，即可查找子串的位置

String 对象处理

◆子字符串与字符查找

```
char[] charArr = { ' ', 'J', 'a', 'v', 'a', ' ' };
str1 = new String(charArr);  // 使用字符数组
str2 = new String("程序语言的程序设计");
// 查找子字符串和字符
System.out.print("英-字符indexOf(\'a\', 2): ");
System.out.println(str1.indexOf('a', 2));
System.out.print("英-字符lastIndexOf(\'b\',2): ");
System.out.println(str1.lastIndexOf('b', 2));
System.out.print("中-字符串indexOf(\"语言\"): ");
System.out.println(str2.indexOf("语言"));
System.out.print("中-字符串lastIndexOf(\"语言\"):");
System.out.println(str2.lastIndexOf("语言"));
```

String 对象处理

◆子字符串与字符处理

方法	说明
char charAt(int)	取得参数int索引位置的字符
String substring(int)	从参数int的位置开始取出剩下的子字符串
String substring(int, int)	取出第一个参数（包括）到第二个参数（不包括）之间的字符串
String replace(char, char)	将字符串中找到的第一个参数char替换为第二个参数
String concate(String)	将参数字符串添加到String对象字符串之后并返回（不影响原字符串）
String trim()	删除字符串前后的空格符

String 对象处理

◆子字符串与字符处理

```
char[] charArr = { ' ', 'J', 'a', 'v', 'a', ' ' };
str1 = new String(charArr);  // 使用字符数组
str2 = new String("程序语言的程序设计");
// 子字符串与字符处理
System.out.print("英文str1.charAt(4): ");
System.out.println(str1.charAt(3));
System.out.print("中文str2.substring(2, 6): ");
System.out.println(str2.substring(2, 6));
System.out.print("替换-英str1.replace('a','b'):");
System.out.println(str1.replace('a','b'));
System.out.print("删除空格符str1.trim(): ");
System.out.println(str1.trim());
String str0 = str1.concat(str2); // 连接两字符串
System.out.println("连接str1.concat(str2): "+str0);
```

String 对象处理

◆字符串比较

String 对象的字符串可以一个一个字符地比较字符的内码值，直到分出大小为止。

方法	说明
int compareTo(String)	比较两个字符串的内容，返回值是整数，0表示相等，<0表示传入参数的字符串比较大，>0表示传入参数的字符串比较小
int compareToIgnoreCase(String)	忽略大小写，比较两个字符串的大小
boolean equals(Object)	比较两个对象是否相等
boolean equalsIgnoreCase(String)	忽略大小写，比较两个字符串是否相等
boolean endsWith(String)	判断字符串是否以传入参数为结尾
boolean startsWith(String)	判断字符串是否以传入参数为开头

String 对象处理

◆字符串比较

```
char[] charArr = { ' ', 'J', 'a', 'v', 'a', ' ' };
str1 = new String(charArr);
str4 = "use";
String str = " JAVA ";
// 显示字符串str和str1的比较结果
System.out.print("比较str与str1字符串: ");
System.out.println(str.compareTo(str1));
System.out.print("比较str与str1字符串-不分大小写: ");
System.out.println(str.compareToIgnoreCase(str1)); // 检查字符串的字头和
字尾
System.out.print("str4的结尾是否为\"s\": ");
System.out.println(str4.endsWith("s"));
System.out.print("str4的字头是否为\"u\": ");
System.out.println(str4.startsWith("u"));
```

String 的切割

◆字符串切割

① String 对象的字符串如果包含确定的分隔符，则可以用split() 方法进行分割

“.”和“|”都是转义字符，必须得加"\\";

```
String str = “www.android.com";
String[] temp;
String  delimeter = “\\."; // 指定分割字符
temp = str.split(delimeter); // 分割字符串
```

② StringTokenizer切割

```
String str = "www.android.com";
String[] temp;
StringTokenizer str2=new StringTokenizer(str, “.");

// 对 str2 遍历并打印子字符串；
  while(str2.hasMoreTokens()){
     System.out.println(str2.nextToken());
  }
```

正则表达式

https://blog.csdn.net/xqclll/article/details/60757392?utm_medium=distribute.pc_relevant.none-task-blog-title-1&spm=1001.2101.3001.4242

将字符串转换成数值

- 可以使用Byte、Short、Integer、Long、Double和Float类的类方法来将字符串内容转换成各种基本数据类型数值。

方法	说明
byte Byte.parseByte(String)	将String转换为byte型基本数据
short Short.parseShort(String)	将String转换为short型基本数据
int Integer.parseInt(String)	将String转换为int型基本数据
long Long.parseLong(String)	将String转换为long型基本数据
float Float.parseFloat(String)	将String转换为float型基本数据
double Double.parseDouble(String)	将String转换为double型基本数据

将字符串转换成数值（例子）

```
// 使用parse ()方法将字符串转换成数值
byte   num1 = Byte.parseByte("5");
short  num2 = Short.parseShort("100");
int    num3 = Integer.parseInt("2000");
long   num4 = Long.parseLong("135");
float  num5 = Float.parseFloat("245.675");
double num6 = Double.parseDouble("145.67891234");
```

将数值转换为字符串

- 可以使用Byte、Short、Integer、Long、Double和Float类的类方法来将字符串内容转换成各种基本数据类型数值。

方法	说明
String Byte.toString(byte)	将byte型基本数据转换为String
String Short.toString(short)	将short型基本数据转换为String
String Integer.toString(int)	将int型基本数据转换为String
String Long.toString(long)	将long型基本数据转换为String
String Float.toString(byte)	将float型基本数据转换为String
String Double.toString(byte)	将double型基本数据转换为String

将数值转换为字符串（例子）

```
// 变量声明
byte num1 = 5;
short num2 = 20;
int num3 = 200;
long num4 = 12000;
float num5 = 234.56f;
double num6 = 124.566789;
// 使用类方法toString()转换成字符串
String str1 = Byte.toString(num1);
String str2 = Short.toString(num2);
String str3 = Integer.toString(num3);
String str4 = Long.toString(num4);
String str5 = Float.toString(num5);
String str6 = Double.toString(num6);
```

课堂练习(1)

- 给定N=100，利用等差数列求和公式，输出前N个自然数的和。
- 给定a=1，b=3，c=-4，输出一元二次方程$ax^2+bx+c=0$的两个解（求平方根可用 Math.sqrt()）。
- 生成一个3*3的二维数组，各元素为随机整数（范围1~100）（随机数可用Math.random()）：

  （1）输出每一行和每一列的总和；

  （2）如果每一行的总和均大于100，输出“yes”，否则输出“no”；

  （3）输出由每一列的总和直接拼成的字符串（如结果依次为25，38，156，则输出2538156）。

清華大學 Tsinghua University

Java文件IO

File对象

- Java的标准库java.io提供了File对象来操作文件和目录。
- 要构造一个File对象，需要传入文件路径：

```
import java.io.*;
public class Main {
    public static void main(String[] args) {
        File f = new File("C:\\Windows\\notepad.exe");
        System.out.println(f);
    }
}
```

File对象

- 构造File对象时，既可以传入绝对路径，也可以传入相对路径。
- 绝对路径是以根目录开头的完整路径，例如：
  - File f = **new** File("C:\\Windows\\notepad.exe");
  - Windows平台使用\作为路径分隔符，在Java字符串中需要用\\表示一个\。
- 传入相对路径时，相对路径前面加上当前目录就是绝对路径：
  - // 假设当前目录是C:\Docs
  - File f1 = new File("sub\\javac"); // 绝对路径是C:\Docs\sub\javac
  - File f2 = new File(".\\sub\\javac"); // 绝对路径是C:\Docs\sub\javac
  - File f3 = new File("..\\sub\\javac"); // 绝对路径是C:\sub\javac

File对象

- File对象既可以表示文件，也可以表示目录。特别要注意的是，构造一个File对象，即使传入的文件或目录不存在，代码也不会出错，因为构造一个File对象，并不会导致任何磁盘操作。只有当我们调用File对象的某些方法的时候，才真正进行磁盘操作。

调用isExist()，判断文件是否存在。

调用createNewFile()，新建文件。

调用delete()，删除文件

调用isFile()，判断是否是一个已存在的文件

调用isDirectory()，判断该是否是一个已存在的目录

调用canRead()，是否可读

调用canWrite()，是否可写

调用canExecute()，是否可执行

调用length()，字节文件大小

File对象

- 当File对象表示一个文件时，可以通过createNewFile()创建一个新文件，用delete()删除该文件：

```
import java.io.*;
public class File_test3 {
    public static void main(String[] args) throws IOException {
    File f = new File("haha1");
    boolean k = f.createNewFile();
    System.out.println(k);
    System.out.println(f.delete());
    }
}
```

文本文件的读写

```
File fp = new File("aaa.txt");
if(!fp.exists()){
        fp.createNewFile();
}
FileWriter fw = new FileWriter(fp);
fw.write("aaa");
fw.write("bbb\n");
fw.write("ccc");
fw.write("ddd");
fw.close();
```

```
FileWriter fileWriter = new FileWriter(fp);
BufferedWriter Writer = new
        BufferedWriter(fileWriter);
Writer.write("aaa1");
Writer.write("bbb2\n");
Writer.write("ccc3");
Writer.write("ddd4");
Writer.close();
fileWriter.close();
```

```
File fp = new File("E:\\aaa.txt");

FileReader fileReader = new
        FileReader(fp);
BufferedReader bufferedReader = new
        BufferedReader(fileReader);
String s1 =
bufferedReader.readLine();
for(;s1 != null;s1 =
bufferedReader.readLine()){
        System.out.println(s1);
}
```

二进制文件的读写

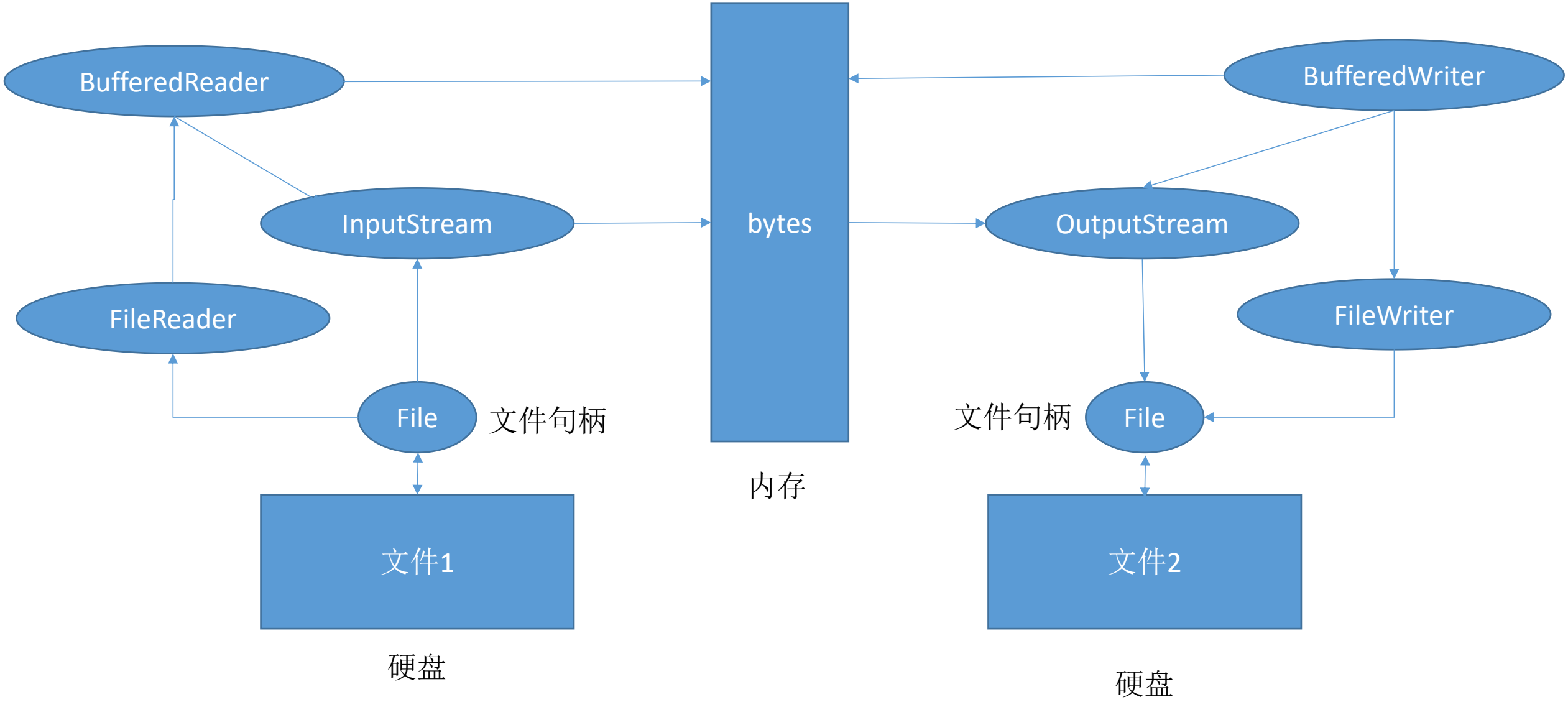
文件读

```
File fp = new File("E:\\日内瓦发明展金奖-2022.pdf");
System.out.println((int)fp.length());
byte[] buffer = new byte[(int)fp.length()];
InputStream input = new FileInputStream(fp))
int n = input.read(buffer);
```

文件写

```
fp = new File("E:\\\\日内瓦发明展金奖-2022.new.pdf");
if(!fp.exists()){
        fp.createNewFile();
}
OutputStream output = new FileOutputStream(fp);
output.write(buffer);
output.close();
```

文件的读写


BufferedReader
BufferedWriter
InputStream
bytes
OutputStream
FileReader
FileWriter
File
文件句柄
文件句柄
File
内存
文件1
文件2
硬盘
硬盘

InputStream

- InputStream就是Java标准库提供的最基本的输入流。它位于java.io这个包里。

  下面的代码演示了如何完整地读取一个FileInputStream的所有字节：

```
import java.io.*;
public class File_test4 {
    public static void main(String[] args) throws IOException {
        InputStream input = new FileInputStream("compare.py");
        for (;;) {
            int n = input.read(); // 反复调用read()方法，直到返回-1
            if (n == -1) {
            break;
            }
        System.out.println(n); // 打印byte的值
        }
        input.close(); // 关闭流
    }
}
```

这里的n是文件字节的ASCII码！

InputStream

- 仔细观察上面的代码，会发现一个潜在的问题：如果读取过程中发生了IO错误，InputStream就没法正确地关闭。因此，我们需要用try 来保证InputStream在无论是否发生IO错误的时候都能够正确地关闭：

```
import java.io.*;
public class File_test4 {
    public static void main(String[] args) throws IOException {
        try (InputStream input = new FileInputStream("compare.py")) {
            int n;
            while ((n = input.read()) != -1) {
                System.out.println(n);
            }
        }
    }
}
```

InputStream

- 在读取流的时候，一次读取一个字节并不是最高效的方法。很多流支持一次性读取多个字节到缓冲区，对于文件和网络流来说，利用缓冲区一次性读取多个字节效率往往要高很多。
  - int read(byte[] b)：读取若干字节并填充到byte[]数组，返回读取的字节数

```
import java.io.*;
public class File_test {
    public static void main(String[] args) throws IOException {
    try (InputStream input = new FileInputStream("readme.txt")) {
        // 定义1000个字节大小的缓冲区:
        byte[] buffer = new byte[1000];
        int n;
        while ((n = input.read(buffer)) != -1) { // 读取到缓冲区
            System.out.println("read " + n + " bytes.");
            String str = new String(buffer);
            System.out.println(str);
        }
    }
}
```

OutputStream

- 和InputStream相反，OutputStream是Java标准库提供的最基本的输出流。

```
public static void main() throws IOException {
    OutputStream output = new FileOutputStream("out/readme.txt");
    output.write(72); // H
    output.write(101); // e
    output.write(108); // l
    output.write(108); // l
    output.write(111); // o
    output.close();
}
```

OutputStream

- 每次写入一个字节非常麻烦，更常见的方法是一次性写入若干字节。

  这时，可以用OutputStream提供的重载方法void write(byte[])来实现：

```
public static void main() throws IOException {
    OutputStream output = new FileOutputStream("out/readme.txt");
    output.write("Hello".getBytes("UTF-8")); // Hello
    output.close();
}
```

OutputStream

- 和InputStream一样，上述代码没有考虑到在发生异常的情况下如何正确地关闭资源。写入过程也会经常发生IO错误，例如，磁盘已满，无权限写入等等。我们需要用try来保证OutputStream在无论是否发生IO错误的时候都能够正确地关闭：

```
public static void main() throws IOException {
    try (OutputStream output = new FileOutputStream("out/readme.txt")) {
        output.write("Hello".getBytes("UTF-8")); // Hello
    }
}
```

课堂练习（2）

1. 请利用InputStream和OutputStream，编写一个复制文件的程序，它可以带参数运行：
   - java CopyFile.java source.txt copy.txt

清華大學 Tsinghua University

Java集合变量

Java中的集合类

◆为什么出现集合类?

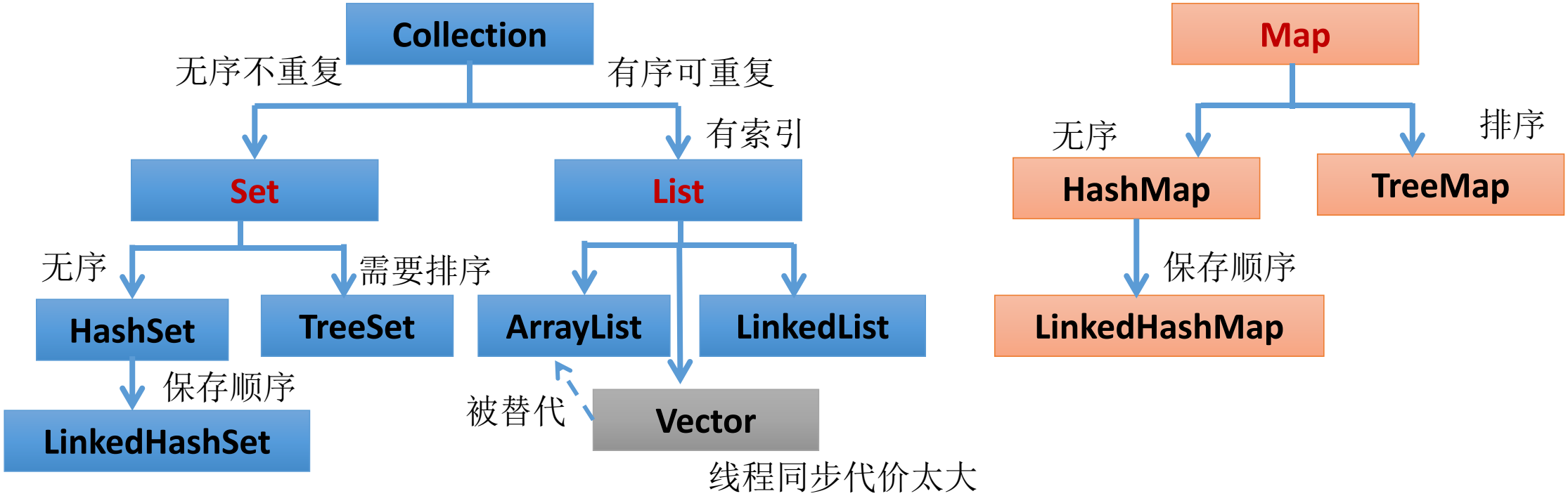
- 在Java中，如果一个Java对象可以在内部持有若干其他Java对象，并对外提供访问接口，我们把这种Java对象称为集合。很显然，Java的数组可以看作是一种集合。
- 既然Java提供了数组这种数据类型，可以充当集合，那么，我们为什么还需要其他集合类？这是因为数组有如下限制：
  - 数组初始化后大小不可变；
  - 数组只能按索引顺序存取。

◆集合类的特点

- 集合长度是可变的，集合可以存储不同类型的对象。
- Java的集合类定义在java.util包中（import java.util.*）

Java 中的集合类概述

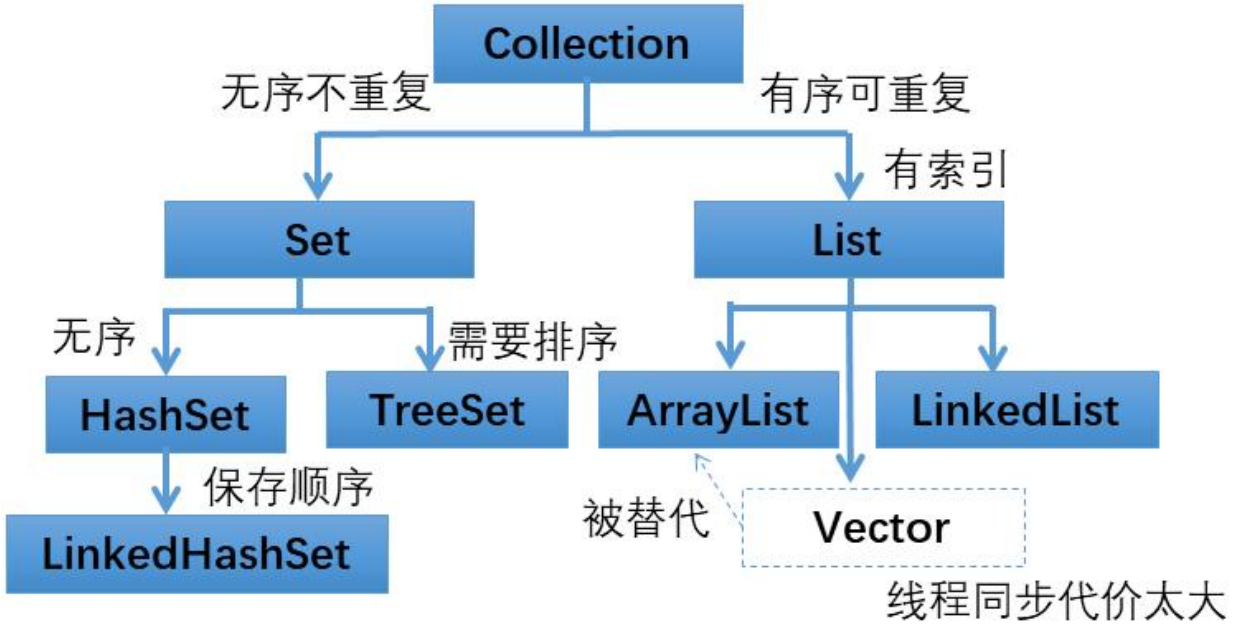
◆Collection 和 Map是Java 集合类的两个根接口，它们派生出了多种集合类



Collection 接口

◆Collection接口有两个子接口：

- List（列表）：可存放重复元素，元素存取是有序的。
- Set（集）：不可以存放重复元素，元素存取是无序的。



List接口中常用类

- Vector：线程安全，但速度慢。
- ArrayList：线程不安全，查询速度快。
- LinkedList：链表结构，增删速度快。
- 取出List集合中元素的方式：
  - get(int index)：通过脚标获取元素。
  - iterator()：通过迭代方法获取迭代器对象。

ArrayList 类

- ArrayList类实现了List接口，顾名思义它使用类似数组的方式来保存元素。
- 元素按照索引位置依次存入，我们不需要事先声明ArrayList的长度，相当于自动调节长度的动态数组

```
import java.util.ArrayList;
import java.util.List;
public class Main {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        list.add("apple"); // size=1
        list.add("pear"); // size=2
        list.add("apple"); // 允许重复添加元素，size=3
        System.out.println(list.size());
    }
}
```

ArrayList 类

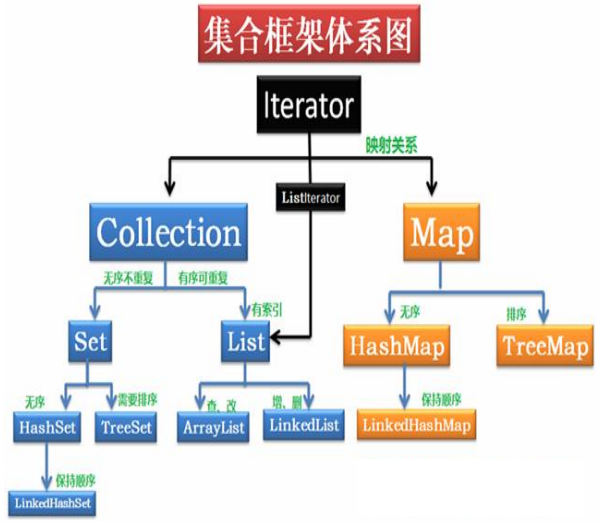
在末尾添加一个元素：boolean add(E e)
在指定索引添加一个元素：boolean add(int index, E e)
删除指定索引的元素：E remove(int index)
删除某个元素：boolean remove(Object e)
获取指定索引的元素：E get(int index)
获取链表大小（包含元素的个数）： int size()

HashSet类

- HashSet类实现了Set接口，是无序的

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Set<String> set = new HashSet<>();
        System.out.println(set.add("abc")); // true
        System.out.println(set.add("xyz")); // true
        System.out.println(set.add("xyz")); // false，添加失败，因为元素已存在
        System.out.println(set.contains("xyz")); // true，元素存在
        System.out.println(set.contains("XYZ")); // false，元素不存在
        System.out.println(set.remove("hello")); // false，删除失败，因为元素不存在
        System.out.println(set.size()); // 2，一共两个元素
    }
}
```



Collections类的静态方法

Collections类对Set、List进行操作，提供的所有方法都是静态的。

- 单元素添加、删除操作
- 排序 sort(list)
  所有元素必须实现Comparable接口
- 随机排序 shuffle(list)
- 移动 swap(list,int m ,int n)
- 数据处理
  - 翻转 reverse(list)
  - 填充 fill(list,Object o)
  - 复制 copy(List m, List n)
- 查找 **indexOfSubList（List m, List m）**
  LastIndexOfSubList（List m, List m）
- 求极值
  - 最小值 min（Collection coll)
  - 最大值 max（Collection coll)
- Collection转换为Object数组 toArray()

```
ArrayList num =  new ArrayList();
num.add(3);
num.add(-1);
num.add(-5);
num.add(10);
System.out.println(num);
System.out.println(Collections.max(num));
System.out.println(Collections.min(num));

Collections.sort(num);
```

课堂练习（2）

有两个列表

- l1 = [11,22,33]
- l2 = [22,33,44]

功能要求：

- 获取内容相同的元素列表
- 获取l1中有，l2中没有的元素列表
- 获取l2中有，l1中没有的元素列表
- 获取l1和l2中内容都不同的元素

课堂练习（3）

- 随机生成100个整数（0~100）之间，添加到List中，并进行一下操作
  - 排序
  - 求最大、最小值
  - 求平均值

```
import java.util.Random;
Random rd = new Random();
for(int i=0;i<50;i++){
    int temp = rd.nextInt(100);
    System.out.println(temp);
}
```

- 给定一组连续的整数，例如：10，11，12，……，20，但其中缺失一个数字，试找出缺失的数字：

课堂练习（4）

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        // 构造从start到end的序列:
        final int start = 10;
        final int end = 20;
        List<Integer> list = new ArrayList<>();
        for (int i = start; i <= end; i++) {
            list.add(i);
        }
        // 随机删除List中的一个元素:
        int removed = list.remove((int) (Math.random() * list.size()));
        int found = findMissingNumber(start, end, list);
        System.out.println(list.toString());
        System.out.println("missing number: " + found);
        System.out.println(removed == found ? "测试成功" : "测试失败");
    }
    static int findMissingNumber(int start, int end, List<Integer> list) {
        return 0;//待修改
    }
}
```

Map接口

- 与List相比，Map 提供了一个更通用的元素存储方法。Map 集合类用于存储元素对（称作“键”和“值”），其中每个键映射到一个值。

Key 1	Value 1
Key 2	Value 2
Key 3	Value 3
…	…
Key n	Value n

Key 与 Value 一一对应，Map中存储的数据，Key是唯一的（不能重复），Value没有这个要求

- 常用的Map包括
  - HashMap
  - Hashtable
  - LinkedHashMap

- 要保证键的唯一性，当键是复杂对象时，需要覆盖hashCode方法，和equals方法（为什么？回忆之前讲过的”==”与equals方法）

HashMap

- HashMap是最常用的Map,它根据Key的HashCode 值存储数据,根据Key可以直接获取它的Value，具有很快的访问速度。HashMap最多只允许一条记录的Key为Null(多条会覆盖);允许多条记录的Value为 Null。HashMap是非同步的。

- HashMap的初始化

```
HashMap<String, String> hashMap = new HashMap<String, String>();
```

- HashMap插入元素

```
map.put("key1", "value1");
```

- HashMap获取元素

```
map.get(“key1”);
```

- HashMap移除元素

```
map.remove("key1");
```

- 清空HashMap

```
map.clear();
```

HashMap

◆ 例程

```
public class Demo1 {
	public static void main(String[] args) {
		// 定义一个Map的容器对象
		HashMap<String, Integer > map1 = new HashMap<String, Integer >();
		map1.put("jack", 20);
		map1.put("rose", 18);
		map1.put("lucy", 17);
		map1.put("jack", 25);
		 // 添加重复的键值（值不同）,后者会覆盖前者
		System.out.println(map1.get(“jack”));

	}
}
```

HashMap的遍历

```
HashMap<String, String > map = new HashMap<String, String >();
map.put("jack", "20");
map.put("rose", "18");
map.put("lucy", "17");

//第一种：普遍使用，二次取值
System.out.println("\n通过Map.keySet遍历key和value：");
for (String key : map.keySet()) {
   System.out.println("key= "+ key + " and value= " + map.get(key));
}
//第二种
System.out.println("\n通过Map.entrySet使用iterator遍历key和value：");
Iterator<Map.Entry<String, String>> it = map.entrySet().iterator();
while (it.hasNext()) {
   Map.Entry<String, String> entry = it.next();
   System.out.println("key= " + entry.getKey() + " and value= " + entry.getValue());
}
//第三种：推荐，尤其是容量大时
System.out.println("\n通过Map.entrySet遍历key和value");
for (Map.Entry<String, String> entry : map.entrySet()) {
    System.out.println("key= " + entry.getKey() + " and value= " + entry.getValue());
}
```

课堂练习(6)

1. 在不改变列表数据结构的情况下找最大值li = [1,3,2,7,6,23,41,243,33,85,56]
2. 在不改变列表中数据排列结构的前提下，找出以下列表中最接近最大值和最小值的平均值的数
   - li = [-100,1,3,2,7,6,120,121,140,23,411,99,243,33,85,56]
3. 查找List/Map中元素，移除每个元素的空格，并查找以a或A开头并且以c结尾的所有元素。
   - li = ["alec", " aric", "Alex", "Tony", "rain"]
   - tu = ("alec", " aric", "Alex", "Tony", "rain")
   - dic = {'k1': "alex", 'k2': ' aric', "k3": "Alex", "k4": "Tony"}
5. 求100以内不能被3整除的所有数，并把这些数字放在列表里，并求出这些数字的总和和平均数。
4. 计算求100以内的质数和。
5. 有四个数字：1、2、3、4，能组成多少个互不相同且无重复数字的三位数？各是多少？

课堂练习(7)

作业：简易购物车
需求分析：
1、启动程序后，在程序中 构建 商品 和 用户 的 对象集合（Map）
2、支持用户选择以下操作代码：（查看商品 (g)、查看余额 (u)、购买商品(b) 、退出购买(q) ）
    2.1、输入q，则退出。
    2.2、输入g，则查看所有商品信息。
    2.3、输入u，则查看用户余额信息。
    2.4、输入b，则进入购买买流程：
        2.4.1、允许用户根据商品名称购买商品。
        2.4.2、用户选择商品 和 商品数量 后，检测余额是否够，够就直接扣款，不够就提醒。

```
goods = [
    {"name": "电脑", "price": 1999},
    {"name": "鼠标", "price": 10},
    {"name": "书籍", "price": 50},
    {"name": "空调", "price": 2000},
]
users = [
    {"name": "张三", 'money': 1000.0},
    {"name": "李四", 'money' :2000.0},
    {"name": "王五", 'money': 2000.0},
]
```