Java编程基础

Outline

- 输入处理输出
- 变量
- 操作符
- 表达式
- 有代码就得有测试
- 决策
- 方法
- 重复
- 数据结构
- 持久化
- 使用API
- String I
- ArrayList
- String II
- 枚举

1. 输入处理输出

编程的典型场景

- 输入
- 处理
- 输出

求两个数和

- 输入
  - 控制台输入2个数
- 处理
  - 将输入String转出int
  - 求和
- 输出
  - 输出和

控制台输入输出

```
public static void twoIntAdd(){
    int one=0;
    int two=0;
    int sum = 0;
    String temp=null;
    try {
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        System.out.println("please enter the first integer:”);
      temp=br.readLine();// 1、输入
      one=Integer.parseInt(temp); // 2.1 处理(转为int)
      System.out.println("please enter the second integer:");
         temp=br.readLine();
      two=Integer.parseInt(temp);
     } catch (IOException e) {
      e.printStackTrace();
    }
     sum = one + two; // 2.2 、处理（求和）
     System.out.println(one+"+"+two+"="+sum);  // 3、输出
}
```

计算

- 变量
- 操作符
- 表达式

2 变量

变量

- Primitive variable 基础数据类型
- Reference variable 引用变量

示例

```
// Compute the area of a circle。

class Area {

    public static void main（String args[]） {

        double pi，r，a;

       r = 10.8; // radius of circle

       pi = 3.1416; // pi，approximately

       a = pi * r * r; // compute area

       System.out.println(“Area of circle is " + a);

    }

}
```

基本数据类型

- 不用其他类型来定义的数据类型被称为基本数据类型。
- 几乎所有程序设计语言都提供一组基本数据类型。
  - 某些基本数据类型仅仅是硬件的反映，例如整数类型。而另一些基本类型只是在实现上需要一点非硬件的支持。

示例

```
// Compute the area of a circle。

class Area {

    public static void main（String args[]） {

        double pi，r，a;  //声明

       r = 10.8; // radius of circle //赋数值给变量

       pi = 3.1416; // pi，approximately

       a = pi * r * r; // compute area//赋变量的值给另一个变量

       System.out.println(“Area of circle is " + a);

    }

}
```

Java基本数据类型

- 整数：
  - 该组包括字节型（byte），短整型（short），整型（int），长整型（long），它们是有符号整数。
- 浮点型数：
  - 该组包括浮点型（float），双精度型（double），它们代表有小数精度要求的数字。
- 字符：
  - 这个组包括字符型（char），它代表字符集的符号，例如字母和数字。
- 布尔型：
  - 这个组包括布尔型（boolean），它是一种特殊的类型，表示真/假值。

- 基础数据类型是其他类型数据的基础。
- Java是完全面向对象的，但简单数据类型不是。
- 因为Java可移植性的要求，所有的数据类型都有一个严格的定义的范围。

整数类型

名称	长度	数的范围
长整型(long)	64	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
整型(int)	32	–2,147,483,648 ~ 2,147,483,647
短整型(short)	16	–32,768 ~ 32,767
字节型(byte)	8	–128~127

浮点类型

名称	位数	数的范围
double	64	1.7E–308～1.7E+308
float	32	3.4E–038～3.4E+038

命名

- 在JAVA中，标识符可以有不同的长度，但是它必须由字母、下划线（_），或者美元符（$）开头，而其余部分可以是除了JAVA运算符（比如+、-或者*）之外的任何字符，但是通常最好只使用字母、数字和下划线字符。
- Java是区分大小写
- 在名字中间不能包括空格或者制表符

常量

- 类常量的声明，应该全部大写，单词间用下划线隔开。例如
- static final int MIN_WIDTH = 4;
- static final int MAX_WIDTH = 999;
- static final int GET_THE_CPU = 1;

3 操作符

操作符

- Assignment
- Arithmetic Operators
- Unary Operators
- Equality and Relational Operators
- Conditional Operators
- Bitwise and Bit Shift Operators

简单赋值操作符

- int cadence = 0;
- int speed = 0;
- int gear = 1;

The Arithmetic Operators

- + additive operator (also used for String concatenation)
- - subtraction operator
- * multiplication operator
- / division operator
  - 如果操作数是整数，得到的是整数
  - 如果操作数有一个是浮点数，另一个会强制转换为浮点数，得到的是浮点数
- % remainder operator

Priority

- 默认优先级
  - * /
  - +-
- ()改变运算顺序

The Unary Operators

- + Unary plus operator; indicates positive value (numbers are positive without this, however)
- - Unary minus operator; negates an expression
- ++ Increment operator; increments a value by 1
- -- Decrement operator; decrements a value by 1
- ! Logical complement operator; inverts the value of a boolean

Equality and relational operators

- == equal to
- != not equal to
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to

The Conditional Operators

- && Conditional-AND
- || Conditional-OR

Bitwise and bit shift operators

- The unary bitwise "~" inverts a bit pattern; it can be applied to any of the integral types, making every "0" a "1" and every "1" a "0
- The signed left shift operator "<<" shifts a bit pattern to the left,
- The signed right shift operator ">>" shifts a bit pattern to the right.
- The unsigned right shift operator ">>>" shifts a zero into the leftmost position, while the leftmost position after ">>" depends on sign extension.

- The bitwise & performs a bitwise AND operation.
- The bitwise ^ performs a bitwise exclusive OR operation.
- The bitwise | performs a bitwise inclusive OR operation.

4 表达式

表达式（Expressions）

- An expression is a construct made up of variables, operators, and method invocations, which are constructed according to the syntax of the language, that evaluates to a single value.

表达式语句

- The following types of expressions can be made into a statement by terminating the expression with a semicolon (;).
  - Assignment expressions
  - Any use of ++ or --
  - Method invocations
  - Object creation expressions

5 有代码就得有测试！

源代码

- public String kindofTriangle (double side1, double side2, double side3)
- …
- return …;
- }

黑盒测试

黑盒测试

- 不了解程序的内部情况
- 只知道程序的输入、输出和系统的功能

黑盒测试一般流程

- 0. 初始化测试
- 1. 调用被测方法得到实际结果result
- 2. 与期望的结果相比较
- 3. 输出测试的结果

测试代码

- public void testRightTriangle（）{
- //0.初始化测试
- String testResult = “fail”;

- //1. 调用被测方法得到实际结果result
- String result = kindofTriangle(3.0, 4.0, 5.0);
- if(result.equals(“Right Triangle”)){ //2. 与期望的结果（Right Triangle）相比较
- testResult = “pass”; // 改变测试的结果
- }

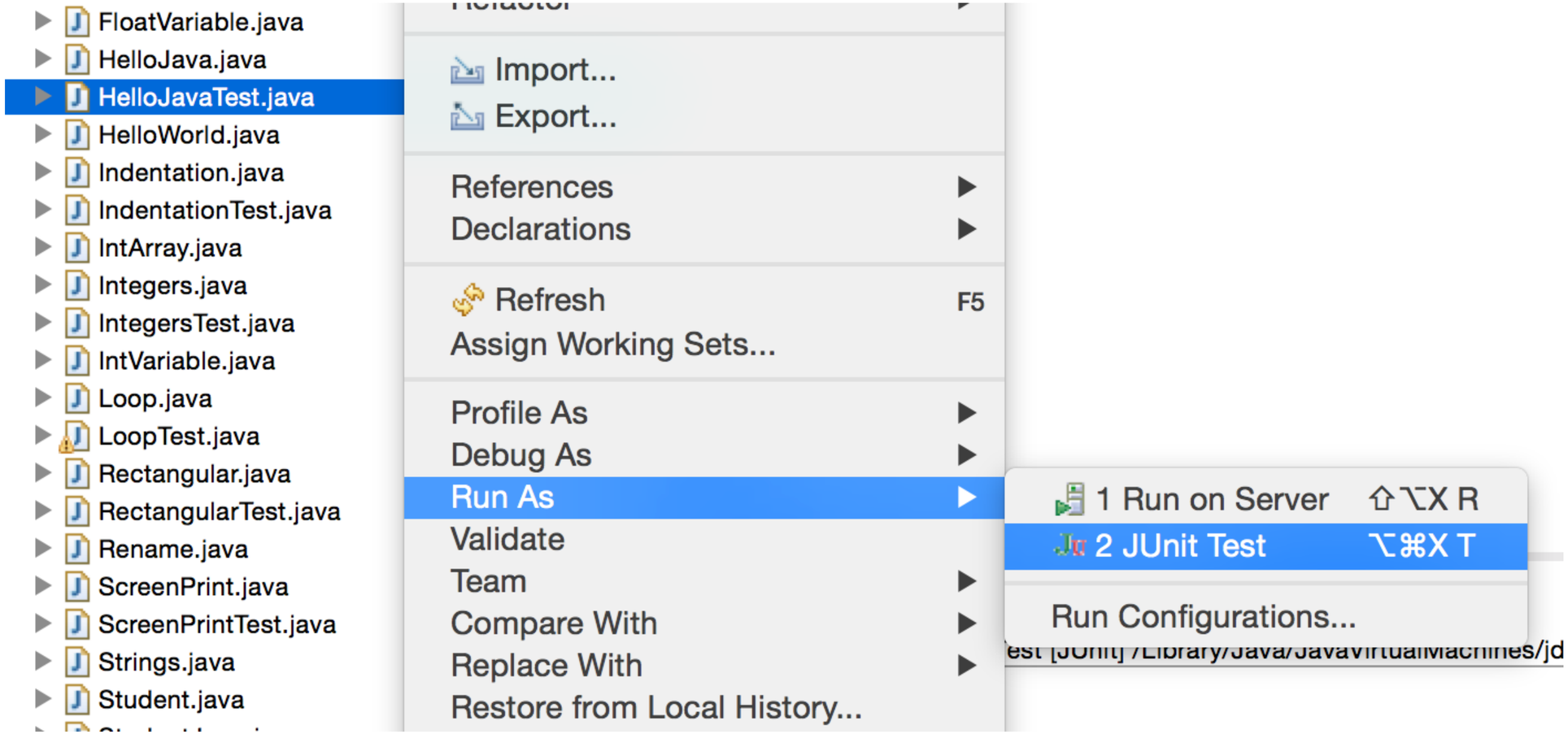
- //3. 输出测试的结果
- System.out.println(testResult);
- }

```java
public class HelloJava {

    public String hellojava() {
        //Submit the code without any revision
        return "Hello Java!";
    }

}
```

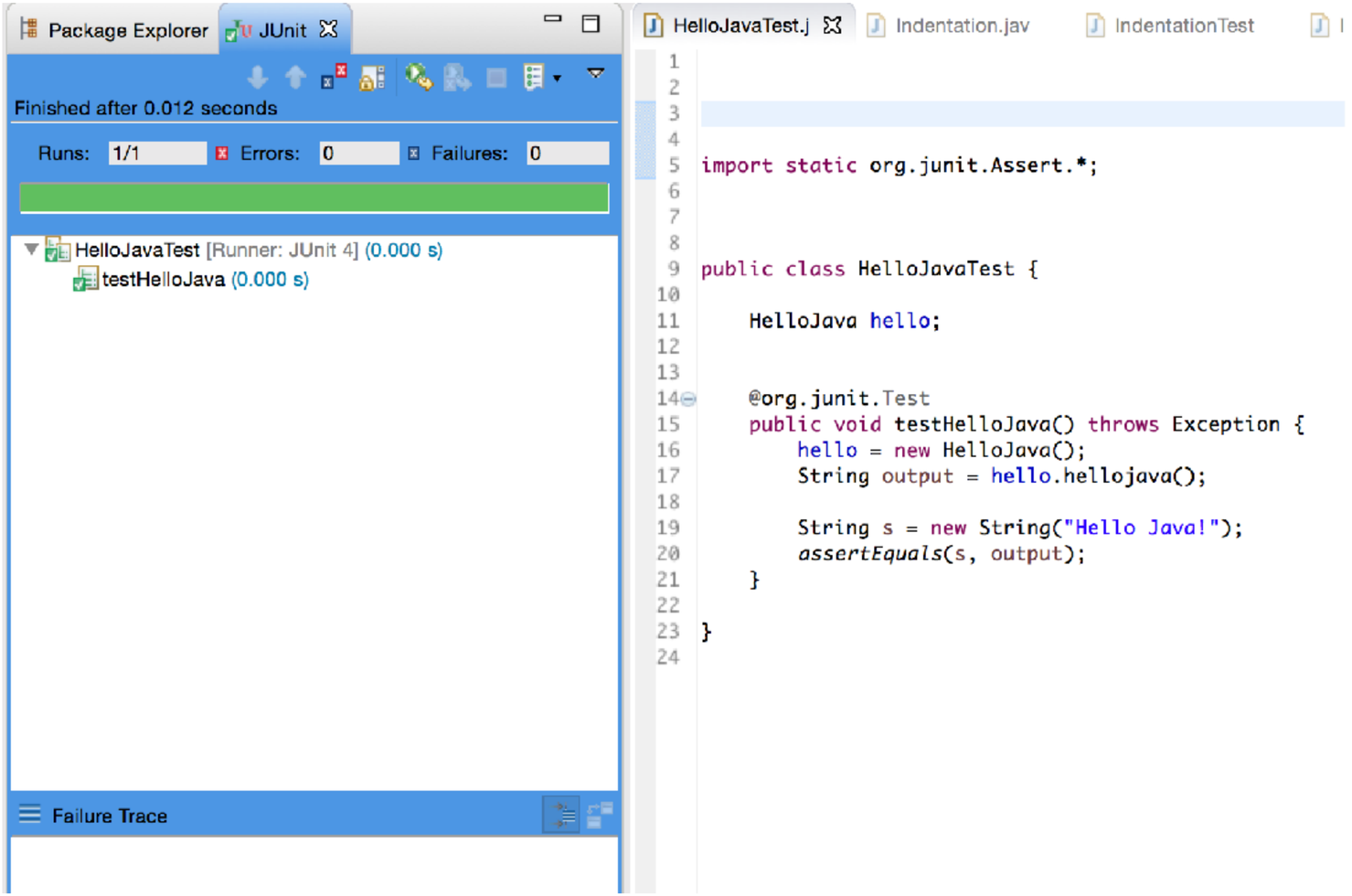
源程序

```java
import java.io.ByteArrayOutputStream;



public class HelloJavaTest {

    HelloJava hello;


    @org.junit.Test
    public void testHelloJava() throws Exception {
        hello = new HelloJava();
        String output = hello.hellojava();

        String s = new String("Hello Java!");
        assertEquals(s, output);
    }

}
```

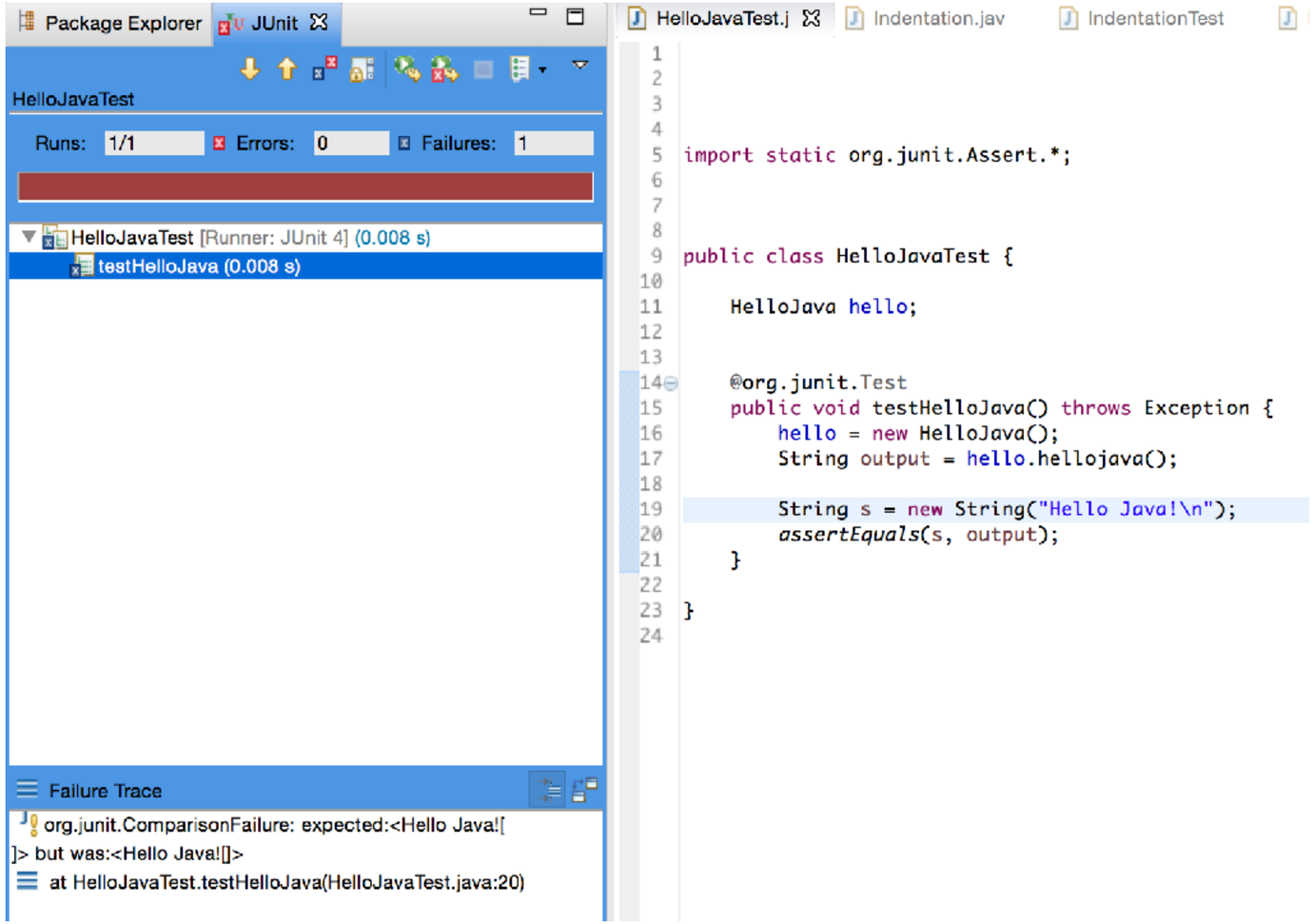
测试程序


FloatVariable.java
HelloJava.java
HelloJavaTest.java
HelloWorld.java
Indentation.java
IndentationTest.java
IntArray.java
Integers.java
IntegersTest.java
IntVariable.java
Loop.java
LoopTest.java
Rectangular.java
RectangularTest.java
Rename.java
ScreenPrint.java
ScreenPrintTest.java
Strings.java
Student.java
Import...
Export...
References
Declarations
Refresh
F5
Assign Working Sets...
Profile As
Debug As
Run As
Validate
Team
Compare With
Replace With
Restore from Local History...
1 Run on Server
2 JUnit Test
Run Configurations...

运行测试用例


Package Explorer
JUnit
Finished after 0.012 seconds
Runs: 1/1
Errors: 0
Failures: 0
HelloJavaTest [Runner: JUnit 4] (0.000 s)
testHelloJava (0.000 s)
Failure Trace
HelloJavaTest.j
Indentation.jav
IndentationTest
import static org.junit.Assert.*;
public class HelloJavaTest {
HelloJava hello;
@org.junit.Test
public void testHelloJava() throws Exception {
hello = new HelloJava();
String output = hello.hellojava();
String s = new String("Hello Java!");
assertEquals(s, output);
}
}

结果


Package Explorer
JUnit
HelloJavaTest
Runs: 1/1
Errors: 0
Failures: 1
HelloJavaTest [Runner: JUnit 4] (0.008 s)
testHelloJava (0.008 s)
Failure Trace
org.junit.ComparisonFailure: expected:<Hello Java![
]> but was:<Hello Java![]>
at HelloJavaTest.testHelloJava(HelloJavaTest.java:20)
HelloJavaTest.j
Indentation.jav
IndentationTest
import static org.junit.Assert.*;
public class HelloJavaTest {
HelloJava hello;
@org.junit.Test
public void testHelloJava() throws Exception {
hello = new HelloJava();
String output = hello.hellojava();
String s = new String("Hello Java!\n");
assertEquals(s, output);
}
}

错误的结果

Homework 1

- Deadline: March 15 23:59:59
- Tasks:
  - 编写一个货币兑换程序，将欧元兑换成美元。请输入手中的欧元数，以及欧元的当前汇率。打印可以兑换的美元数。
  - 示例输出：
  - How many eruos are you exchanging? 81
  - What is the exchange rate? 137.51
  - 81 euros at an exchange rate of 137.51 is
  - 111.38 U.S. dollars.
- Submit:
  - 源代码

6 决策

决策

- 选择场景
  - 条件
  - 分支

if else

```
if(Boolean-expression)
  statement
```

or

```
if(Boolean-expression)
  statement
else
  statement
```

```
//: control/VowelsAndConsonants.java
// Demonstrates the switch statement.
import java.util.*;
import static net.mindview.util.Print.*;

public class VowelsAndConsonants {
  public static void main(String[] args) {
    Random rand = new Random(47);
    for(int i = 0; i < 100; i++) {
      int c = rand.nextInt(26) + 'a';
      printnb((char)c + ", " + c + ": ");
      switch(c) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u': print("vowel");
                  break;
        case 'y':
        case 'w': print("Sometimes a vowel");
                  break;
        default:  print("consonant");
      }
    }
  }
} /* Output:
y, 121: Sometimes a vowel
n, 110: consonant
z, 122: consonant
b, 98: consonant
r, 114: consonant
n, 110: consonant
y, 121: Sometimes a vowel
g, 103: consonant
c, 99: consonant
f, 102: consonant
o, 111: vowel
w, 119: Sometimes a vowel
z, 122: consonant
...
*///:~
```

switch

Java SE 7 新特性

```
public class StringSwitchDemo {

    public static int getMonthNumber(String month) {

        int monthNumber = 0;

        if (month == null) {
            return monthNumber;
        }

        switch (month.toLowerCase()) {
            case "january":
                monthNumber = 1;
                break;
            case "february":
                monthNumber = 2;
                break;
            case "march":
                monthNumber = 3;
                break
            …
            case “december”:
                monthNumber = 12;
                break;
            default:
                monthNumber = 0;
                break;
        }// end of switch
        return monthNumber;
    }// end of getMonthNumber()
    …
}// end of class StringSwitchDemo
```

为什么要有变量?

- 变量的空间可以复用
  - 空间维度上，不同代码使用同一个变量
  - 时间维度上，同一代码不同时间上使用这个变量
  - 值变化的维度，变量自身值是会变化
- 10个程序分别求1到10的和、平方和、立方和。。。10次方和

7 方法（函数）

为什么要方法调用?

如果没有方法调用?

方法（函数）的意义

- 逻辑的封装
- 重用（空间维度）
- 可修改（时间维度）

方法被调用

- 方法被调用的特性
  - 每个方法都只有一个入口。
  - 当执行被调用的方法的时候，调用方法暂停。
  - 当方法结束时，程序的控制权交还给调用处。

例子

```
public class HelloWorld {
        int f(int i){
                return i;
        }
         public static void main(String[] args) {
                HelloWorld hw = new HelloWorld();
                int n = 5;
                 int value = hw.f(n);
                System.out.println("Value = "+ value);
        }
}
```

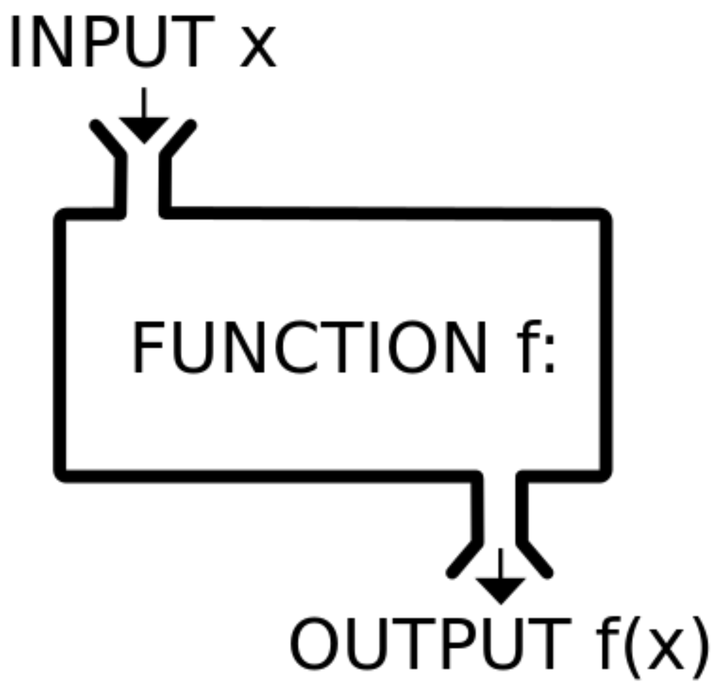
形参（parameter）与实参（argument）

- 形参
  - A parameter is the variable which is part of the method’s signature (method declaration).
- 实参
  - An argument is an expression used when calling the method.

例子

```
public class HelloWorld {

    int f(int i){    // i是形参

        return i;

    }

    public static void main(String[] args) {

        HelloWorld hw = new HelloWorld();

        int n = 5;

        int value = hw.f(n); //n 是实参

        System.out.println("Value = "+ value);

    }

}
```

返回值



返回值

- 三者保持一致
  - 返回值本身的类型
  - 返回值的类型
  - 返回之后赋值的类型
- int f(int n){
  - return n ;
- }
- int i =5
- int j = f(i)

多参数

- 一一对应
  - f(int a, int b)
  - f(2,3)

Homework 2

- Deadline: March 15, 23:59:59

- 编写一个程序，让用户输入3个数。首先确认所有数字各不相同，如果存在相同的数，退出程序，否则显示其中最大的。

- 示例输出
- Enter the first number: 1
- Enter the second number: 51
- Enter the third number: 2
- The largest number is 51.

- Submit:
    - 源代码

8 重复

典型场景

- 计数器
  - 提示用户输入5个数字，计算他们的和
- 直到条件满足
  - 当用户输入键入无效的输入值，不要退出，而是不断提示用户输入，直到获得有效的输入值
- 递归
  - 通过递归实现循环
- 嵌套循环
  - 乘法表

流程图

开始

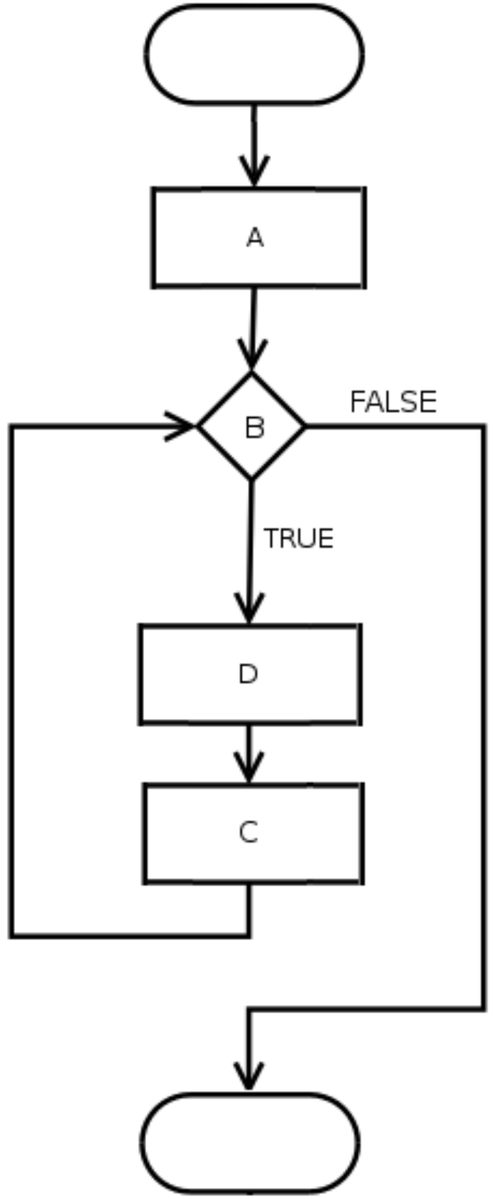
结束

操作

条件

流

```
for(A;B;C)
  D;
```



Iteration

```
while(Boolean-expression)
  statement
```

```
do
  statement
while(Boolean-expression);
```

```
for(initialization; Boolean-expression; step)
  statement
```

foreach syntax

```
//: control/ForEachFloat.java
import java.util.*;

public class ForEachFloat {
  public static void main(String[] args) {
    Random rand = new Random(47);
    float f[] = new float[10];
    for(int i = 0; i < 10; i++)
      f[i] = rand.nextFloat();
    for(float x : f)
      System.out.println(x);
  }
} /* Output:
//: control/ForEachString.java

public class ForEachString {
  public static void main(String[] args) {
    for(char c : "An African Swallow".toCharArray() )
      System.out.print(c + " ");
  }
} /* Output:
A n   A f r i c a n   S w a l l o w
*///:~
```

break & continue

- break
  - breaks out of the inner iteration and you end up in the outer iteration.
  - break label (breaks all the way out to label, but it does not reenter the iteration)
- continue
  - the continue moves back to the beginning of the inner iteration.
  - continue label (reenter the iteration)

return

- Specifies what value a method will return
- Causes the current method to exit

Infamous goto

- 1968
- Edsger W. Dijkstra
- A Case against the GO TO Statement

Homework-03

- Deadline: March 22, 23:59:59
- 卡蒙内心率
  - TargetHeartRate =(((220-age)-RestingHR)*intensity)+RestingHR
- 示例输出
  - Resting Pulse: 65
  - Age: 22 //提示输入
  - Intensity | Rate
  - ---------|------
  - 55% |138bpm
  - 60% |145bpm
  - 。。。
  - 95% |191bpm
- Submit:
  - 源码

9 数据结构

数据结构

- 数据的组织
  - 数组
  - 列表
  - 哈希
  - 映射
  -

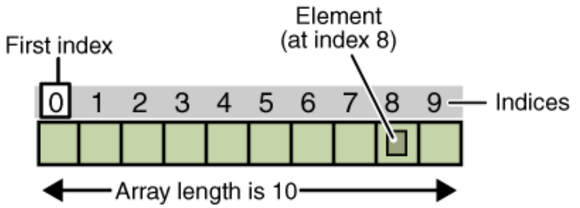
数据结构往往和循环一起使用

典型场景

- 打印一组姓名中的每一个
- 根据随机数，随机从答案数组选一个
- 从员工列表中删除一个元素
- 抽奖
- 计算统计信息
- 过滤值
- 排序记录

Array

- Creating
  - int[] anArray; // declares an array of integers
  - char[] copyTo = new char[7]; // create a array object
- Initializing
  - anArray[0] = 100; // initialize first element
  - int[] anArray = {100, 200, 300, 400, 500, 600, 700, 800, 900, 1000};
- Accessing
  - System.out.println("Element 1 at index 0: " + anArray[0]);
- Length
  - System.out.println(anArray.length);
- Copying
  - public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)



Copy数组

```
class ArrayCopyDemo {
    public static void main(String[] args) {
        char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e',
                            'i', 'n', 'a', 't', 'e', 'd' };
        char[] copyTo = new char[7];

        System.arraycopy(copyFrom, 2, copyTo, 0, 7);
        System.out.println(new String(copyTo));
    }
}
```

Copy数组

```
class ArrayCopyOfDemo {
    public static void main(String[] args) {

        char[] copyFrom = {'d', 'e', 'c', 'a', 'f', 'f', 'e',
            'i', 'n', 'a', 't', 'e', 'd'};

        char[] copyTo = java.util.Arrays.copyOfRange(copyFrom, 2, 9);

        System.out.println(new String(copyTo));
    }
}
```

二维数组

```
class MultiDimArrayDemo {
    public static void main(String[] args) {
        String[][] names = {
            {"Mr. ", "Mrs. ", "Ms. "},
            {"Smith", "Jones"}
        };
        // Mr. Smith
        System.out.println(names[0][0] + names[1][0]);
        // Ms. Jones
        System.out.println(names[0][2] + names[1][1]);
    }
}
```

ArrayList	regular array
`ArrayList<String> myList = new ArrayList<String>();`	`String [] myList = new String[2];`
`String a = new String("whoohoo");`	`String a = new String("whoohoo");`
`myList.add(a);`	`myList[0] = a;`
`String b = new String("Frog");`	`String b = new String("Frog");`
`myList.add(b);`	`myList[1] = b;`
`int theSize = myList.size();`	`int theSize = myList.length;`
`Object o = myList.get(1);`	`String o = myList[1];`
`myList.remove(1);`	`myList[1] = null;`

`boolean isIn = myList.contains(b);`	`boolean isIn = false;`
	`for (String item : myList) {`
	`if (b.equals(item)) {`
	`isIn = true;`
	`break;`
	`}`
	`}`

Here's where it starts to look really different...

ArrayList

Homework-4

- Deadline: March 22, 23:59:59
- 编写一个程序，提示输入某个网站的响应时间，以毫秒表示，不断让用户输入值，直到用户输入“done”。改程序应打印平均时间（mean），最小时间（min），最大时间（max）和标准差（standard deviation）
- 示例输出
    - Enter a number: 100
    - Enter a number: 200
    - Enter a number: 1000
    - Enter a number: 300
    - Enter a number: done
    - Numbers: 100,200,1000,300
    - The average is 400.
    - The minimum is 100.
    - The maximum is 1000.
    - The standard deviation is 400.25.
- Submit:
    - 源码

10 持久化

持久化

- 文件
- 对象序列化
- 网络

文件输入输出

- File
- FileWriter & FileReader
- BufferWriter & BufferReader

Java.io.File类

Some things you can do with a File object:

① Make a File object representing an existing file

```
File f = new File("MyCode.txt");
```

② Make a new directory

```
File dir = new File("Chapter7");
dir.mkdir();
```

③ List the contents of a directory

```
if (dir.isDirectory()) {
      String[] dirContents = dir.list();
      for (int i = 0; i < dirContents.length; i++) {
          System.out.println(dirContents[i]);
      }
}
```

④ Get the absolute path of a file or directory

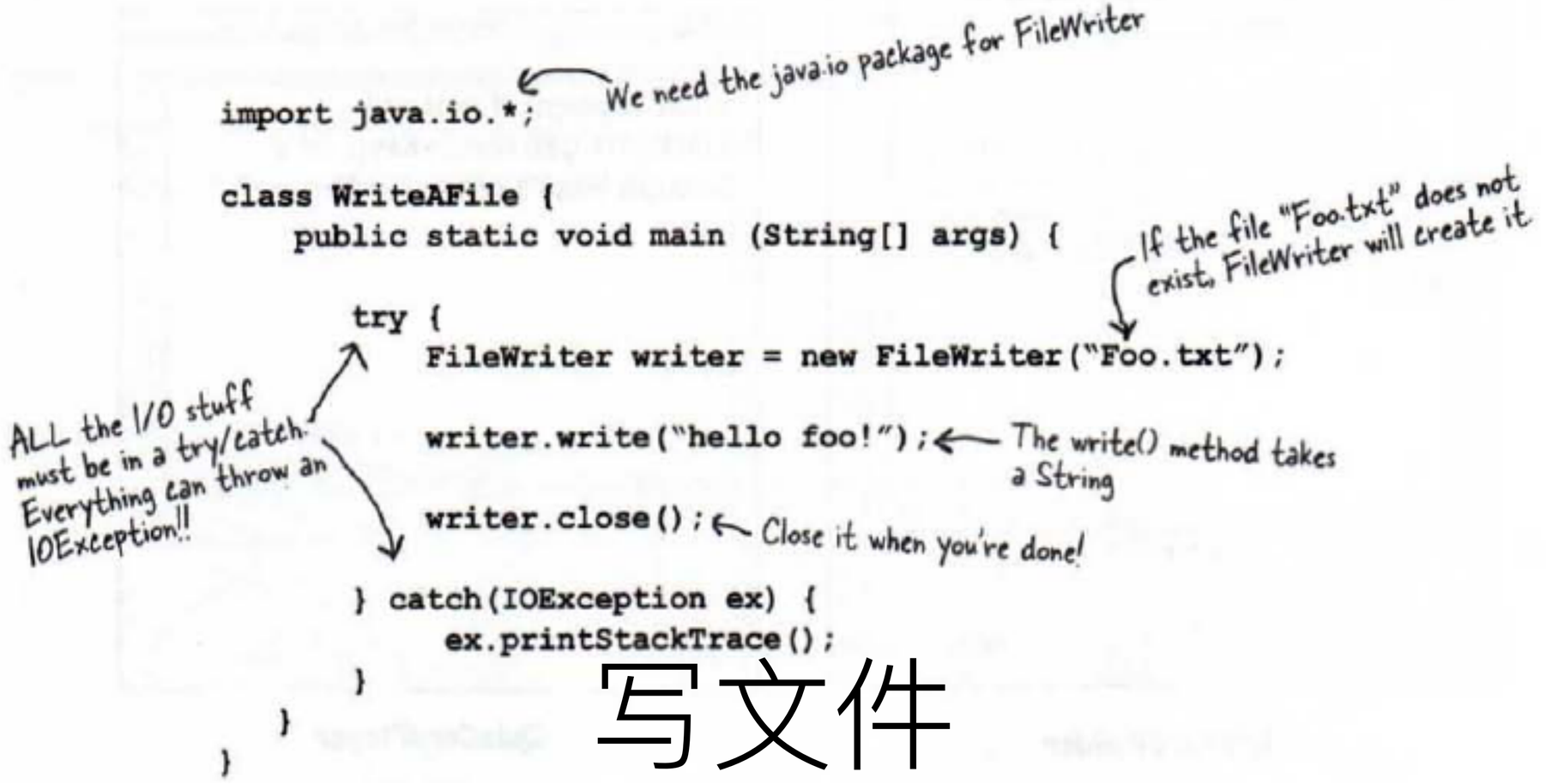
```
System.out.println(dir.getAbsolutePath());
```

⑤ Delete a file or directory (returns true if successful)

```
boolean isDeleted = f.delete();
```

To write a String:

```
fileWriter.write("My first String to save");
```



写文件

```
class ReadAFile {
    public static void main (String[] args) {

        try {
            File myFile = new File("MyText.txt");
            FileReader fileReader = new FileReader(myFile);

            BufferedReader reader = new BufferedReader(fileReader);

            String line = null;

            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();

        } catch(Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

A FileReader is a connection stream for characters, that connects to a text file

Chain the FileReader to a BufferedReader for more efficient reading. It'll go back to the file to read only when the buffer is empty (because the program has read everything in it).

Make a String variable to hold each line as the line is read

This says, "Read a line of text, and assign it to the String variable 'line'. While that variable is not null (because there WAS something to read) print out the line that was just read."

Or another way of saying it, "While there are still lines to read, read them and print them."

读文件

要了解文件读写模式，需要了解几种模式的区别，以及对应指针

r：读取文件，若文件不存在则会报错

w: 写入文件，若文件不存在则会先创建再写入，会覆盖原文件

a：写入文件，若文件不存在则会先创建再写入，但不会覆盖原文件，而是追加在文件末尾

rb,wb：分别于r,w类似，但是用于读写二进制文件

r+：可读、可写，文件不存在也会报错，写操作时会覆盖

w+：可读，可写，文件不存在先创建，会覆盖

a+：可读、可写，文件不存在先创建，不会覆盖，追加在末尾

文件读写模式

CSV文件

- 逗号分隔值（Comma-Separated Values，CSV，有时也称为字符分隔值，因为分隔字符也可以不是逗号），其文件以纯文本形式存储表格数据（数字和文本）。纯文本意味着该文件是一个字符序列，不含必须象二进制数字那样被解读的数据。CSV文件由任意数目的记录组成，记录间以某种换行符分隔；每条记录由字段组成，字段间的分隔符是其它字符或字符串，最常见的是逗号或制表符。通常，所有记录都有完全相同的字段序列。

- CSV文件格式的通用标准并不存在，但是在RFC 4180中有基础性的描述。使用的字符编码同样没有被指定，但是7-bit ASCII是最基本的通用编码。

样例

- Year,Make,Model,Description,Price
- 1997,Ford,E350,"ac, abs, moon",3000.00
- 1999,Chevy,"Venture ""Extended Edition""","",4900.00
- 1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
- 1996,Jeep,Grand Cherokee,"MUST SELL!
- air, moon roof, loaded",4799.00

年份	品牌	型号	描述	价格
1997	Ford	E350	ac, abs, moon	3000.00
1999	Chevy	Venture "Extended Edition"		4900.00
1999	Chevy	Venture "Extended Edition, Very Large"		5000.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded	4799.00

Homework 5 - Stream

- Deadline: March 29, 23:59:59
- 假如给定一个名称列表，其中一些名称包含一个字符。系统会要求您在一个逗号分隔的字符串中返回名称，该字符串中不包含单字母的名称，每个名称的首字母都大写。
  - 输入List("neal", "s", "stu", "j", "rich", “bob")
  - 输出”Neal,Stu,Rich,Bob”
- 分别用命令式范式和函数式范式实现。
- 提交源代码。

Homework6 - CSVFile

- Deadline: March 29, 23:59:59
- 编写一个程序，读入以下数据文件
- Ling,Mai,55900
- Johnson,Jim,56500
- …
- Zarnecki,Sabrina,51500
- 处理改记录，并以格式化的表格形式显示结果，间隔均匀（4个空格），如示例输出。
  - Last Fisrt Salary
  - Ling Mai 55900
  - Johnson Jim 56500
  - …
  - Zarnecki Sabrina 51500
- 提交源代码。

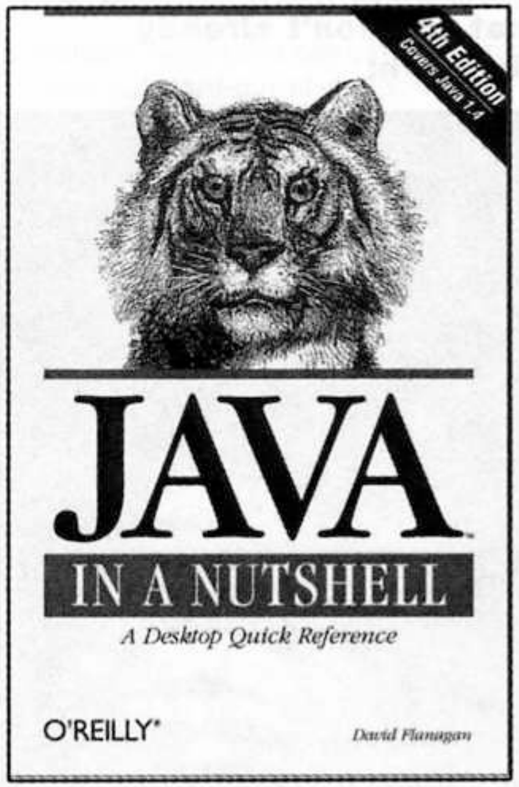
11 使用API

How to play with the API

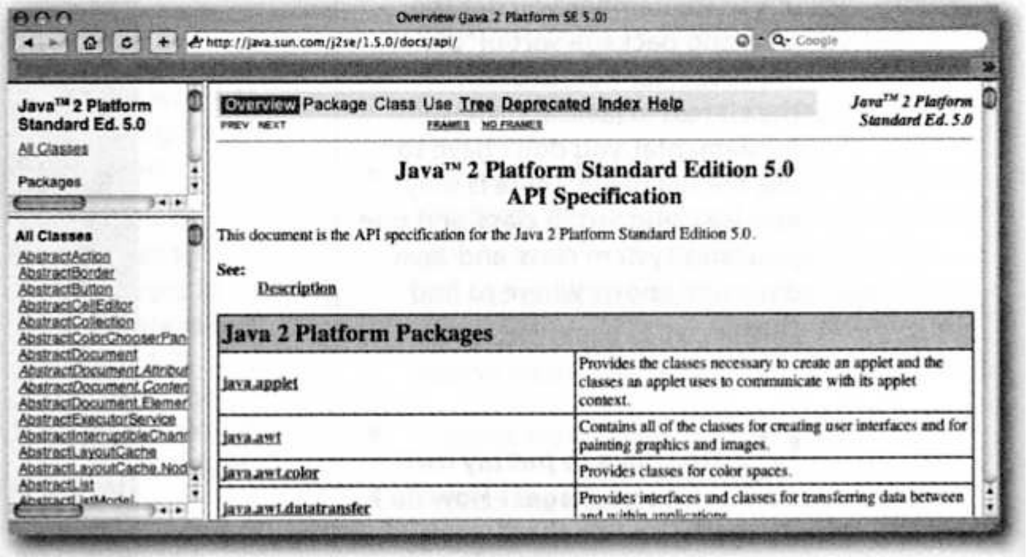
Two things you want to know:

1. **What classes are in the library?**
2. **Once you find a class, how do you know what it can do?**

1 Browse a Book



2 Use the HTML API docs



java.lang

- Object – the class that is the root of every class hierarchy.
- Enum – the base class for enumeration classes.
- Class – the class that is the root of the Java reflection system.
- Throwable – the class that is the base class of the exception class hierarchy.
- Error, Exception, and RuntimeException – the base classes for each exception type.
- Thread – the class that allows operations on threads.
- String – the class for strings and string literals.

java.lang

- StringBuffer and StringBuilder – classes for performing string manipulation.
- Comparable – the interface that allows generic comparison and ordering of objects.
- Iterable – the interface that allows generic iteration using the enhanced for loop.
- ClassLoader, Process, Runtime,SecurityManager, and System – classes that provide "system operations" that manage the dynamic loading of classes, creation of external processes, host environment inquiries such as the time of day, and enforcement of security policies.
- Math and StrictMath – classes that provide basic math functions such as sine, cosine, and square root.
- The primitive wrapper classes that encapsulate primitive types as objects.
- The basic exception classes thrown for language-level and other common exceptions.

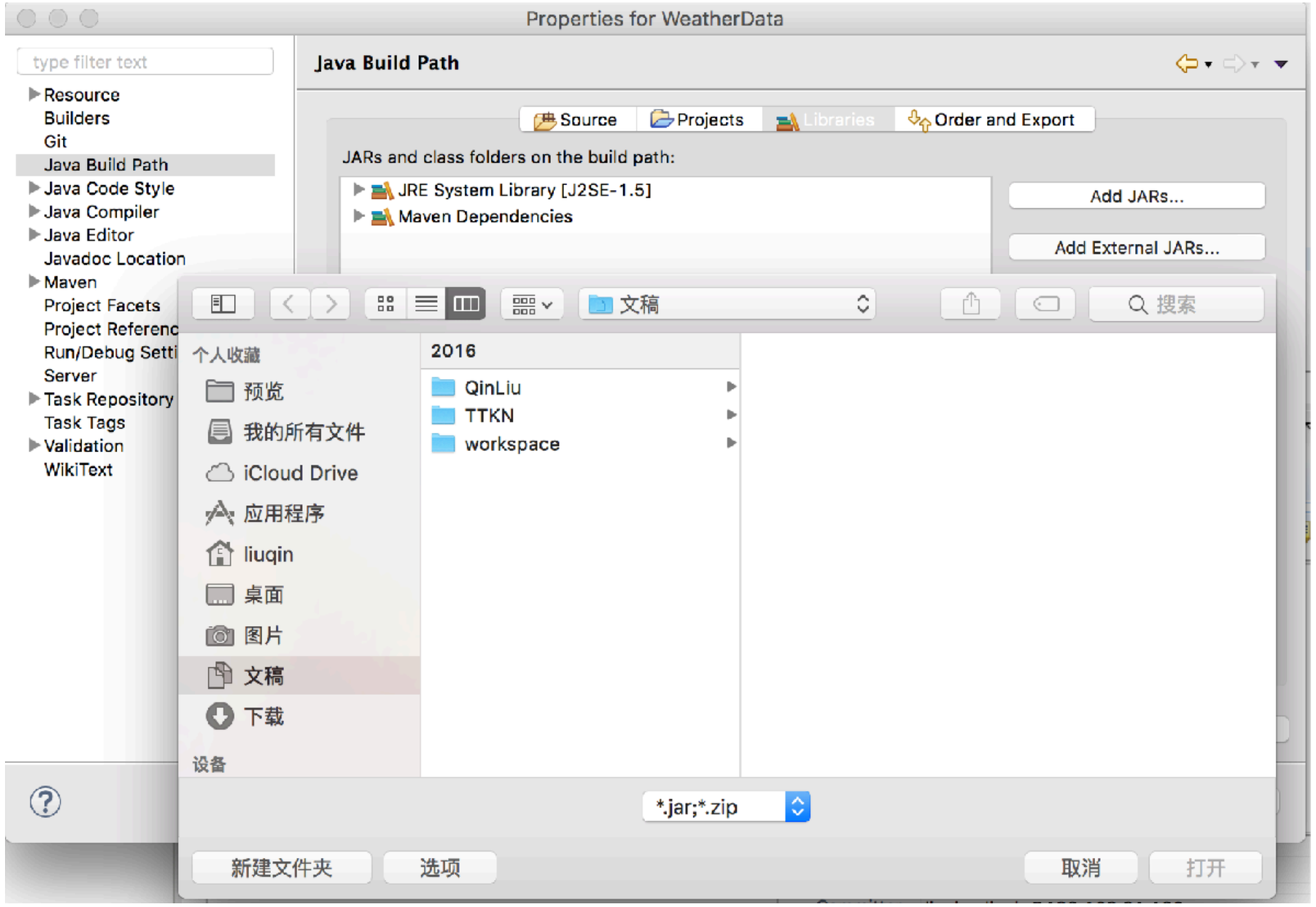
Other Package

- General Purpose Packages
  - Java.io
  - Java.math
  - Java.net
  - Java.text
  - Java.util
- Special Purpose Packages
  - Java.applet
  - Java.beans
  - Java.awt
  - Java.rmi
  - Java.sql
  - Javax.swing

import

- The types that comprise a package are known as the package members.
- To use a public package member from outside its package, you must do one of the following:
  - Refer to the member by its fully qualified name
    - graphics.Rectangle myRect = new graphics.Rectangle();
  - Import the package member
    - import graphics.Rectangle;
    - Rectangle myRectangle = new Rectangle();
  - Import the member's entire package
    - import graphics.*;
    - Circle myCircle = new Circle();
    - Rectangle myRectangle = new Rectangle();

Add External JARs



我们的好朋友
Maven

pom.xml

- <repository>
- <id> homework-maven-repo</id>
- <name> Homework Maven Repo</name>
- <url> http://218.94.159.101:8081/repository/homework-central/</url>
- <releases>
- <enabled>true</enabled>
- </releases>
- <snapshots>
- <enabled>true</enabled>
- </snapshots>
- </repository>
- <dependency>
- <groupId>edu.nju.software</groupId>
- <artifactId>HeWeather</artifactId>
- <version>1.0.0-RELEASE</version>
- </dependency>

Homework 7 - Weather

- Deadline: April 5, 23:59:59

- 1 利用已有API抓取显示当地的天气数据。
  - 1. 获取当前城市名称
  - 2. 根据城市名称获取天气数据
- 2 数据格式如下：
  - {"city":"南京","update_time":"2017-03-27 22:51","weather":"多云","temperature":13,"pressure":1020,"humidity":53}
- 说明：
  - update_time：天气更新时间
  - temperature：温度，单位摄氏度（°C）
  - pressure：气压，单位百帕（mb）
  - humidity：相对湿度，单位百分比（%）

Homework 7 - Weather

- 3. 解析2中获取的天气数据，并按示例格式输出
  - 城市：南京
  - 更新时间：2017-03-27 22:51
  - 天气：多云
  - 温度：13°C
  - 气压：1020mb
  - 相对湿度：53%

API

- String edu.nju.software.HeWeather.getCurrentCity() throws NetworkException
- getCurrentCity
  public String getCurrentCity()
- throws NetworkException
- 获取当前所在城市
- 返回:
  - 当前所在城市名称，如 南京
- 抛出:
  - NetworkException - 网络异常

API

- String edu.nju.software.HeWeather.getWeatherInfo(String city) throws NetworkException
- getWeatherInfo
  public String getWeatherInfo(String city)
- throws NetworkException
- 获取某个城市的天气信息
- 参数:
  - city - 城市名称，如 南京
- 返回:
  - 天气信息的Json字符串
- 抛出:
  - NetworkException - 网络异常

12 String I

String

- 1、String类是final的，不可被继承。public final class String。
- 2、String类是的本质是字符数组char[], 并且其值不可改变。private final char value[];
- 3、String类对象有个特殊的创建的方式，就是直接指定比如String x = "abc"，"abc"就表示一个字符串对象。而x是"abc"对象的地址，也叫做"abc"对象的引用。
- 4、String对象可以通过“+”串联。串联后会生成新的字符串。也可以通过concat()来串联。
- 5、创建字符串的方式很多，归纳起来有三类：
  - 其一，使用new关键字创建字符串，比如String s1 = new String("abc");
  - 其二，直接指定。比如String s2 = "abc";
  - 其三，使用串联生成新的字符串。比如String s3 = "ab" + "c";

substring

- public String substring(int beginIndex, int endIndex)

- Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the length of the substring is endIndex-beginIndex.

- Examples:

- "hamburger".substring(4, 8) returns "urge"

- "smiles".substring(1, 5) returns "mile"

split

- public String[] split(String regex)
- Splits this string around matches of the given regular expression.
- This method works as if by invoking the two-argument split method with the given expression and a limit argument of zero. Trailing empty strings are therefore not included in the resulting array.

- The string "boo:and:foo", for example, yields the following results with these expressions:

- Regex Result
- : { "boo", "and", "foo" }
- o { "b", "", ":and:f" }

比较相等

- == and != compare object reference
- equals()
  - Integer compare the content
  - Your own class unless you override it, compare object reference

案例 1

```
public class StringTest1{
  public static void main(String[] args) {
     String str1 = "abc";
     String str2 = "abc";
     System.out.println(str1 == str2);
   }
}
```

案例 2

```
public class StringTest2 {
    public static void main(String[] args) {
        String str1 = new String("abc");
        String str2 = new String("abc");
        System.out.println(str1 == str2);
    }
}
```

13 ArrayList

ArrayList I

```
public class ArrayListTest {

    public static void main(String[] args) {

        // 创建ArrayList
        ArrayList list = new ArrayList();

        // 将“”
        list.add("1");
        list.add("2");
        list.add("3");
        list.add("4");
        // 将下面的元素添加到第1个位置
        list.add(0, "5");

        // 获取第1个元素
        System.out.println("the first element is: "+ list.get(0));
        // 删除“3”
        list.remove("3");
        // 获取ArrayList的大小
        System.out.println("Arraylist size=: "+ list.size());
        // 判断list中是否包含"3"
        System.out.println("ArrayList contains 3 is: "+ list.contains(“3”));
        // 设置第2个元素为10
        list.set(1, "10");
```

ArrayList II

```
        // 通过Iterator遍历ArrayList
        for(Iterator iter = list.iterator(); iter.hasNext(); )
        {
            System.out.println("next is: "+ iter.next());
        }

        // 将ArrayList转换为数组
        String[] arr = (String[])list.toArray(new String[0]);
        for (String str:arr)
            System.out.println("str: "+ str);

        // 清空ArrayList
        list.clear();
        // 判断ArrayList是否为空
        System.out.println("ArrayList is empty: "+ list.isEmpty());
    }
}
```

运行结果

- the first element is: 5
- Arraylist size=: 4
- ArrayList contains 3 is: false
- next is: 5
- next is: 10
- next is: 2
- next is: 4
- str: 5
- str: 10
- str: 2
- str: 4
- ArrayList is empty: true

14 String II

比较相等

- == and != compare object reference
- equals()
  - Integer compare the content
  - Your own class unless you override it, compare object reference

案例 1

```
public class StringTest1{
  public static void main(String[] args) {
     String str1 = "abc";
     String str2 = "abc";
     System.out.println(str1 == str2);
   }
}
```

案例 2

```
public class StringTest2 {
    public static void main(String[] args) {
        String str1 = new String("abc");
        String str2 = new String("abc");
        System.out.println(str1 == str2);
    }
}
```

String Length

```
public class StringDemo {
    public static void main(String[] args) {
        String palindrome = "Dot saw I was Tod";
        int len = palindrome.length();
        char[] tempCharArray = new char[len];
        char[] charArray = new char[len];

        // put original string in an
        // array of chars
        for (int i = 0; i < len; i++) {
            tempCharArray[i] =
                palindrome.charAt(i);
        }

        // reverse array of chars
        for (int j = 0; j < len; j++) {
            charArray[j] =
                tempCharArray[len - 1 - j];
        }

        String reversePalindrome =
            new String(charArray);
        System.out.println(reversePalindrome);
    }
}
```

Format String

- Using String's static format() method allows you to create a formatted string that you can reuse, as opposed to a one-time print statement. For example, instead of

- System.out.printf("The value of the float " +
- "variable is %f, while " +
- "the value of the " +
- "integer variable is %d, " +
- "and the string is %s",
- floatVar, intVar, stringVar);

- you can write

- String fs;
- fs = String.format("The value of the float " +
- "variable is %f, while " +
- "the value of the " +
- "integer variable is %d, " +
- " and the string is %s",
- floatVar, intVar, stringVar);
- System.out.println(fs);

Convert strings to numbers

- float a = (Float.valueOf(args[0])).floatValue();
- float b = (Float.valueOf(args[1])).floatValue();

- 或者

- float a = Float.parseFloat(args[0]);
- float b = Float.parseFloat(args[1]);

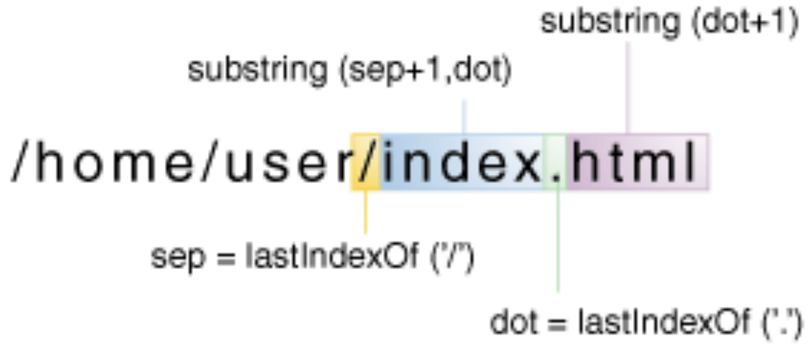
Convert numbers to strings

- int i;
- // Concatenate "i" with an empty string; conversion is handled for you.
- String s1 = "" + i;

- // The valueOf class method.
- String s2 = String.valueOf(i);

- Each of the Number subclasses includes a class method, toString(), that will convert its primitive type to a string. For example:
- int i;
- double d;
- String s3 = Integer.toString(i);
- String s4 = Double.toString(d);

操作字符




```
public class Filename {
    private String fullPath;
    private char pathSeparator,
                 extensionSeparator;

    public Filename(String str, char sep, char ext) {
        fullPath = str;
        pathSeparator = sep;
        extensionSeparator = ext;
    }

    public String extension() {
        int dot = fullPath.lastIndexOf(extensionSeparator);
        return fullPath.substring(dot + 1);
    }

    // gets filename without extension
    public String filename() {
        int dot = fullPath.lastIndexOf(extensionSeparator);
        int sep = fullPath.lastIndexOf(pathSeparator);
        return fullPath.substring(sep + 1, dot);
    }

    public String path() {
        int sep = fullPath.lastIndexOf(pathSeparator);
        return fullPath.substring(0, sep);
    }
}
```

比对

```
public class RegionMatchesDemo {
    public static void main(String[] args) {
        String searchMe = "Green Eggs and Ham";
        String findMe = "Eggs";
        int searchMeLength = searchMe.length();
        int findMeLength = findMe.length();
        boolean foundIt = false;
        for (int i = 0;
            i <= (searchMeLength - findMeLength);
            i++) {
            if (searchMe.regionMatches(i, findMe, 0,
findMeLength)) {
                foundIt = true;
                System.out.println(searchMe.substring(i, i +
findMeLength));
                break;
            }
        }
        if (!foundIt)
            System.out.println("No match found.");
    }
}
```

15 枚举

枚举

- 枚举类型用于表示一组整型常量，用关键字enum来声明，其通用格式如下：
- enum 枚举类型 {
- 枚举值1, 枚举值2, …, 枚举值n
- }
- 枚举值实际上是常量，通常采用大写字母表示，多个枚举值之间用逗号隔开。与直接采用public static final定义常量相比，枚举类型可以保证变量的取值在指定范围内，且枚举值在打印时会直接显示用于标识的字符串而不是实际的整数值。

枚举

- enum LibInterface {
- LOGIN, HOME, BORROWERMANAGE, BOOKMANAGE
- }
- …
- LibInterfance nextInterface;
- …
- switch(nextInterface) {
- case LibInterface.LOGIN: …; break;
- case LibInterface.HOME: …; break; 66

- case LibInterface.BORROWERMANAGE: …; break;
- case LibInterface.BOOKMANAGE: …; break;
- default: …
- }

enum in Java

- The enum declaration defines a class (called an enum type). The enum class body can include methods and other fields. The compiler automatically adds some special methods when it creates an enum.

- For example, they have a static values method that returns an array containing all of the values of the enum in the order they are declared. This method is commonly used in combination with the for-each construct to iterate over the values of an enum type.

```
public enum Planet {
    MERCURY (3.303e+23, 2.4397e6),
    VENUS   (4.869e+24, 6.0518e6),
    EARTH   (5.976e+24, 6.37814e6),
    MARS    (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27,   7.1492e7),
    SATURN  (5.688e+26, 6.0268e7),
    URANUS  (8.686e+25, 2.5559e7),
    NEPTUNE (1.024e+26, 2.4746e7);

    private final double mass;   // in kilograms
    private final double radius; // in meters
    Planet(double mass, double radius) {
        this.mass = mass;
        this.radius = radius;
    }
    private double mass() { return mass; }
    private double radius() { return radius; }

    // universal gravitational constant  (m3 kg-1 s-2)
    public static final double G = 6.67300E-11;

    double surfaceGravity() {
        return G * mass / (radius * radius);
    }
    double surfaceWeight(double otherMass) {
        return otherMass * surfaceGravity();
    }
    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Usage: java Planet <earth_weight>");
            System.exit(-1);
        }
        double earthWeight = Double.parseDouble(args[0]);
        double mass = earthWeight/EARTH.surfaceGravity();
        for (Planet p : Planet.values())
           System.out.printf("Your weight on %s is %f%n",
                             p, p.surfaceWeight(mass));
    }
}
```

values()