Java 面向过程编程

章铁飞　浙江工商大学
tfzhang@mail.zjgsu.edu.cn

1. 变量

2. 逻辑运算发

3. 控制流程

4. 数组相关

5. Java Debug 入门

1. 变量

自测题：

1. java 中的 8 种基本数据类型？
2. 下面的代码有什么错误：

```
public class HelloWorld{
    public static void main(String[] args){
        float f = 12.3;
        boolean a = 0;
        System.out.println("f="+f);
    }
}
```

3. 如下代码中的 a, b 值是多少？c 代表多大的小数？

```
public class HelloWorld{
    public static void main(String[] args){
        int a = 0x1a;
        int b = o32;
        double c = 2.23e1;
    }
}
```

4. 对于属性变量 i，1、2 和 3 处的引用对么？如果有错，请问哪几处对，哪几处错？

```
public class HelloWorld {
      int i = 1;
      int j = i;                //1
      public void method1(){
            int a = i;          //2
      }
      public void method2(){
            System.out.println(i); //3
      }
}
```

5. 对于参数 i 的两处引用 1 和 2，哪几处是对的？哪几处是错的？

```
public class HelloWorld {
   public void method1(int i){
      System.out.println(i);
   }
   public void method2(){
      System.out.println(i); //1
   }

   int j = i;  //2
}
```

6. 如下关于 final 修饰符的使用，哪几处会报错？

a.

```
public class HelloWorld {
   public void method1() {
      final int i = 5;
      i = 10;
   }
}
```

b.

```
public class HelloWorld {
    public void method1() {
        final int i;
        i = 10;
        i = 11;
    }
}
```

c.

```
public class HelloWorld {
```

```
    public void method1(final  int  j) {
        j = 5;
    }
}
```

基本内容：

这八种基本类型分别是：
整型 （4 种）
字符型 （1 种）
浮点型 （2 种）
布尔型（1 种）

整型数据类型

类型	缺省值	长度	数的范围
byte	0	8位	-128 ~ 127
short	0	16位	-32,768 ~ 32,767
int	0	32位	-2,147,483,648 ~ 2,147,483,647
long	0	64位	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

浮点型数据要注意的问题：
浮点数类型有两种
float 长度为 32 位
double 长度为 64 位
注意： 默认的小数值是 double 类型的
所以 float f = 54.321 会出现编译错误，因为 54.321 的默认类型是 double，其类型长度为 64，超过了 float 的长度 32
在数字后面加一个字母 f，直接把该数字声明成 float 类型
float f2 = 54.321f,
这样就不会出错了

浮点型数据类型

类型	缺省值	长度	数的范围
float	0.0	32位	3.4E-038~3.4E+038
double	0.0	64位	1.7E-308~1.7E+308

布尔型数据要注意的问题：
分别代表真假
虽然布尔型真正存放的数据是 0(false) 1(true)
但是，不能直接使用 0 1 进行赋值

类型	缺省值	长度	数的范围
boolean	false	1位	false、true

整数的数制:

```
int hexVal =0x1a; //16 进制
int oxVal =032; //8 进制
```

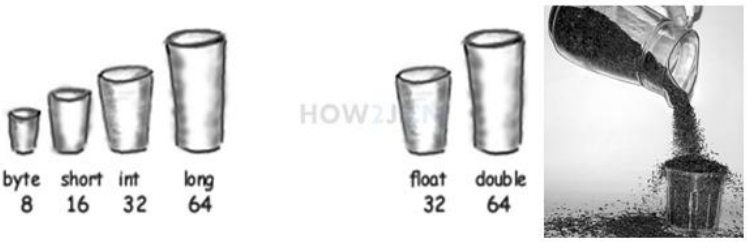
double 数据的表示：

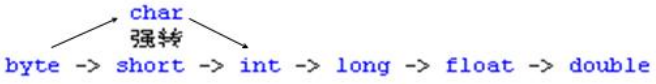
```
double d1 =123.4;// 默认就是 double 类型
double d2 =1.234e2;// 科学计数法表示 double
```

转化规则：
转换规则如图所示
精度高的数据类型就像容量大的杯子，可以放更大的数据
精度低的数据类型就像容量小的杯子，只能放更小的数据
小杯子往大杯子里倒东西，大杯子怎么都放得下
大杯子往小杯子里倒东西，有的时候放的下，有的时候就会有溢出
需要注意的一点是
虽然 short 和 char 都是 16 位的，长度是一样的
但是彼此之间，依然需要进行强制转换。

数据类型转换

规则:从小到大自动转,从大到小强制转.





低精度向高精度转换：

高精度向低精度转化：

作用域：

属性、成员变量、Field：指一个变量被声明在类下面.比如变量 i,就是一个属性。那么从第 2 行这个变量声明的位置开始，整个类都可以访问得到所以其作用域就是从

其声明的位置开始的整个类。

```
public class HelloWorld {
    int i = 1;
    int j = i;   //其他的属性可以访问 i
    public void method1(){
        System.out.println(i); //方法 1 里可以访问 i
    }
    public void method2(){
        System.out.println(i); //方法 2 里可以访问 i
    }
}
```

参数：

指声明在一个方法上的变量。参数的作用域即为该方法内的所有代码，其他方法不能访问该参数，类里面也不能访问该参数.

```
public class HelloWorld {
  public void method1(int i){ //参数 i 的作用域即方法 method1
    System.out.println(i);
  }
  public void method2(){
    System.out.println(i); //method2  不能访问参数 i
  }

  int j = i;  //类里面也不能访问参数 i
}
```

局部变量：

指声明在方法内的变量。其作用域在声明开始的位置，到其所处于的块结束位置。

```
public class HelloWorld {
    public void method1() {
        int i    = 5;   //其作用范围是从声明的第 4 行，到其所处于的块结束 12 行位置
        System.out.println(i);
        {                   //子块
            System.out.println(i); //可以访问 i
            int j = 6;
            System.out.println(j); //可以访问 j
        }
        System.out.println(j); //不能访问 j,因为其作用域到第 10 行就结束了
    }

}
```

final 修饰符：

当一个变量被 final 修饰的时候，该变量只有一次赋值的机会。

```
public class HelloWorld {
   public void method1() {
      final int i = 5;
      i = 10; //i 在第 4 行已经被赋值过了，所以这里会出现编译错误
   }
}
```

上述代码的错误在哪里？

只有一次的声明机会，如下例：

```
public class HelloWorld {

      public void method1() {
            final int i;
            i = 10; //i 在第 4 行，只是被声明，但是没有被赋值，所以在这里可以进行第一次赋值
            i = 11; //i 在第 6 行已经被赋值过了，所以这里会出现编译错误
      }
}
```

如下的代码是否正确：

```
public class HelloWorld {
   public void method1(final int j) {
      j = 5; //这个能否执行？
   }
}
```

2. 逻辑运算符

&和&&， 长路与和短路与的区别：

&和&&， 长路与和短路与的区别：

无论长路与还是短路与两边的运算单元都是布尔值都为真时才为真，任意为假就为假。区别长路与两侧，都会被运算短路与只要第一个是 false，第二个就不进行运算了。

3. 控制流程

练习使用 Eclipse 调试如下程序，找出问题所在：

```
public class HelloWorld {
	public static void main(String[] args) {
		boolean b = false;
		if (b);
			System.out.println("yes");

	}
}
```

4. 数组相关

练习：

1. 补充完成找出最小数的代码

```
public class HelloWorld {
    public static void main(String[] args) {
        int[] a = new int[5];
        a[0] = (int) (Math.random() * 100);
        a[1] = (int) (Math.random() * 100);
        a[2] = (int) (Math.random() * 100);
        a[3] = (int) (Math.random() * 100);
        a[4] = (int) (Math.random() * 100);

        System.out.println("数组中的各个随机数是:");
        for (int i = 0; i < a.length; i++)
            System.out.println(a[i]);

        System.out.println("本练习的目的是，找出最小的一个值: ");
        //将方法写在这里：
    }
}
```

2. 完成数组反转

首先创建一个长度是 5 的数组,并填充随机数。使用 for 循环或者 while 循环，对这个数组实现反转效果。

3. 采用增强型 for 循环找出数组 a（大小为 5）的最大那个数。

4. 采用 System.arraycopy 合并数组。

首先准备两个数组，他俩的长度是 5-10 之间的随机数，并使用随机数初始化这两个数组。然后准备第三个数组，将两个数组合并到第三个数组。

5. 定义一个 5*5 的二维数组，使用随机数填充二维数组。

找出这个二维数组里的最大值，并打印出二维坐标。

1. 如何声明和创建数组：

方法 1：

```
int[] a;
a=new int[5];
```

方法 2：

```
int [] b = new int[5];
```

2. 如何获得数组长度：
a.length

关于增强型 for 循环：

```
public class HelloWorld {
	public static void main(String[] args) {
		int values [] = new int[]{18,62,68,82,65,9};
		//常规遍历
		for (int i = 0; i < values.length; i++) {
			int each = values[i];
			System.out.println(each);
		}

		//增强型 for 循环遍历
		for (int each : values) {
			System.out.println(each);
		}

	}
}
```

如何复制数组？
System.arraycopy(src, srcPos, dest, destPos, length)

src: 源数组
srcPos: 从源数组复制数据的起始位置
dest: 目标数组
destPos: 复制到目标数组的起始位置
length: 复制的长度

```
public class HelloWorld {
	public static void main(String[] args) {
		int a [] = new int[]{18,62,68,82,65,9};
		int b[] = new int[3];//分配了长度是 3 的空间，但是没有赋值

		//通过数组赋值把，a 数组的前 3 位赋值到 b 数组
		//方法一： for 循环
		for (int i = 0; i < b.length; i++) {
			b[i] = a[i];
		}

		//方法二: System.arraycopy(src, srcPos, dest, destPos, length)
		//src: 源数组
		//srcPos: 从源数组复制数据的启始位置
```

```
            //dest: 目标数组
            //destPos: 复制到目标数组的启始位置
            //length: 复制的长度
            System.arraycopy(a, 0, b, 0, 3);

            //把内容打印出来
            for (int i = 0; i < b.length; i++) {
                System.out.print(b[i] + " ");
            }
        }
}
```

如何初始化二维数组？

```
public class HelloWorld {
    public static void main(String[] args) {
        //初始化二维数组，
        int[][] a = new int[2][3]; //有两个一维数组，每个一维数组的长度是 3
        a[1][2] = 5;   //可以直接访问一维数组，因为已经分配了空间

        //只分配了二维数组
        int[][] b = new int[2][]; //有两个一维数组，每个一维数组的长度暂未分配
        b[0]   =new int[3]; //必须事先分配长度，才可以访问
        b[0][2] = 5;

        //指定内容的同时，分配空间
        int[][] c = new int[][]{
                {1,2,4},
                {4,5},
                {6,7,8,9}
        };
    }
}
```

Array 工具类：

数组复制：Arrays.copyOfRange

```
int[] b = Arrays.copyOfRange(a, 0, 3);
```

数组排序：sort(a);

```
Arrays.sort(a);
```

二分查找：binarySearch(int a[], int num)

```
Arrays.binarySearch(a, 62)
```

判断数组是否相同：equals();

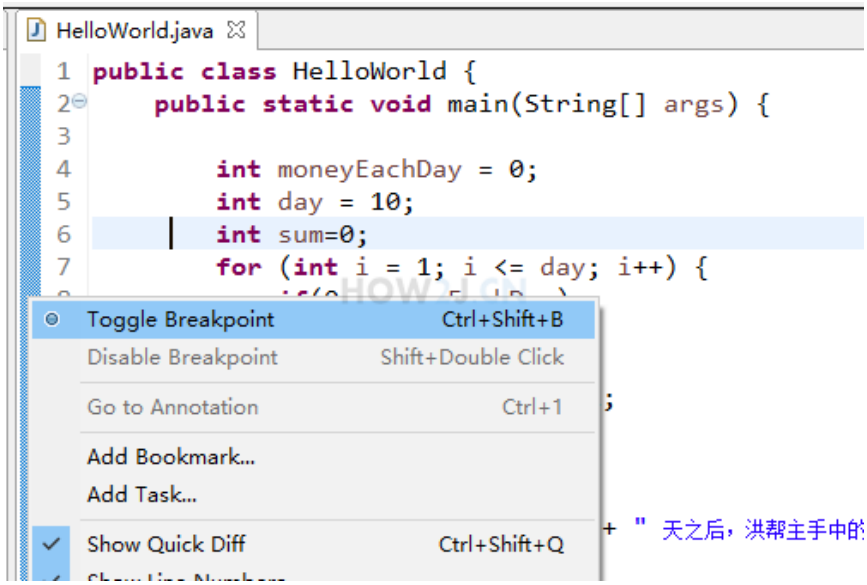
Arrays.equals(a, b);

使用同一个数值填充数组：fill(int a[], int num)
Arrays.fill(a, 5);

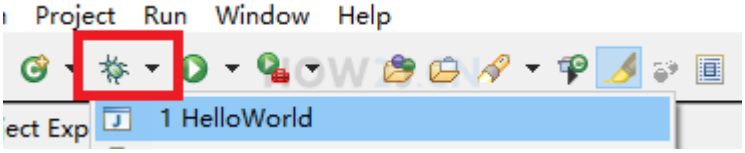
5. Java Debug 流程入门

如何打断点？

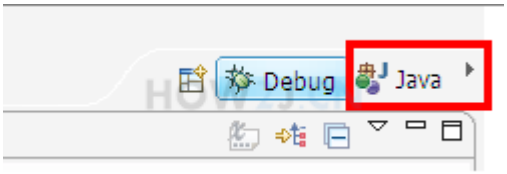
设置断点有几种方式，比如要在第八行设置断点
1. 在行号 8 那个位置，右键点击鼠标，然后选择 Toggle Breakpoint 就可以添加或者删除断点了
2. 在行号 8 那个位置， 双击鼠标左键也可以添加或者删除断点。



如何开始调试？
在平时用运行按钮左边， 有个虫子按钮，就是 debug 按钮。
点击进行调试。

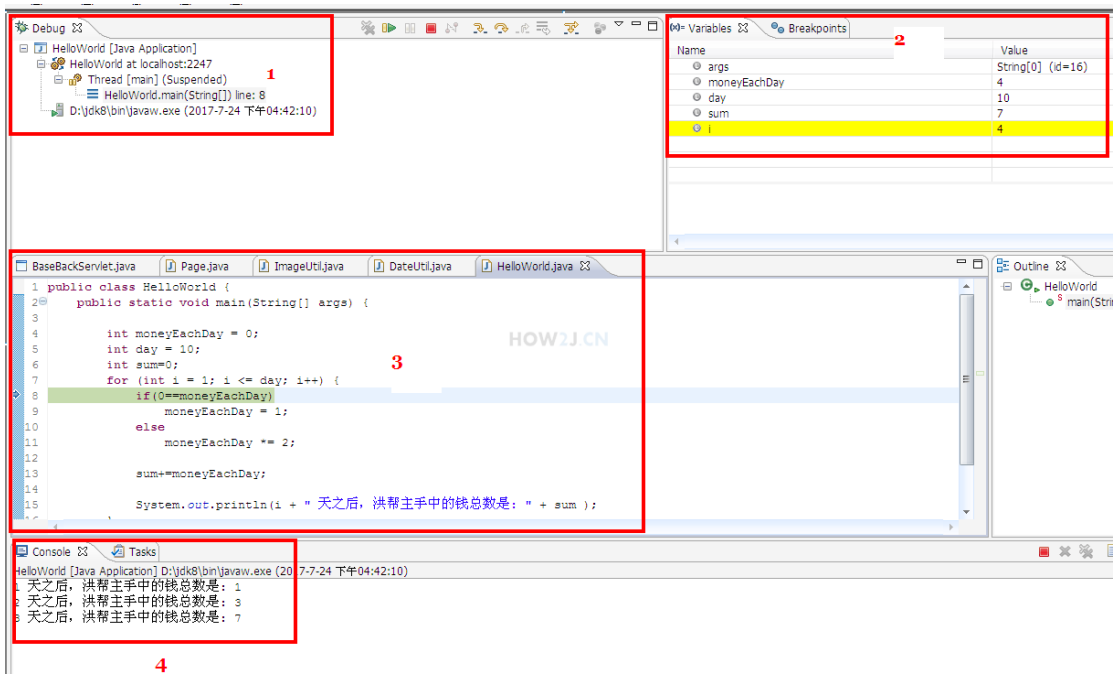


Java 编码时窗口和调试时窗口的切换：



调试时的几个窗口：
1. 当前是哪个线程，因为是非多线程程序，所以就是主线程
2. 对第八行运行有影响的几个变量的值，这个就是调试的主要作用，观察这些值的多少，进行分析问题所在或者理解代码逻辑

3. 当前代码，表示马上就要运行第八行，但是还没有来得及运行第八行
4. 控制台输出



如何单步执行？

如何退出？