Java之helloworld

```
//这是我的HelloWorld案例，class用于定义类
public class HelloWorld {
  /*
   * 这是main方法
   * main方法是程序的入口方法
   * 所有程序的执行都是从main方法开始的
   */
  public static void main(String[] args) {
    //这是输出语句
    System.out.println("HelloWorld");
  }
}
123456789101112
```

Java中的注释方法和标识符

注释用于解释说明代码，分为单行和多行注释，可以提高程序的阅读性。

```
/*
 - 注释：用于解释说明的文字
 - 分类：  
 -     单行
 -     多行
 */
 //这里是单行注释
1234567
```

标识符： 就是用来给包，类，变量，方法等起名字的符号
组成规则：

- unicode字符 数字字符，英文大小写字母，汉字(不建议使用汉字)
- 下划线_
- 美元符$

注意事项
A:不能以数字开头
B:不能是java中的关键字

常量的概念与分类

常量的概念：在程序执行过程中，其值不可以发生改变的量。
分类：
A:字符串常量 “HelloWorld”
B:整数常量 12,23
C:小数常量 12.23
D:字符常量 ‘a’,‘0’
E:布尔常量 true,false
F:空常量

变量的定义和使用

变量的定义：内存中的一小块区域，在程序执行过程中，其值可以在一定范围内发生改变

变量的限制：

1. 对区域要有限制:
   通过数据类型来实现
2. 必须给区域名称:
   变量名
3. 区域内必须有数据
   初始化值

变量的定义格式：
数据类型 变量名 = 初始化值;

注意事项：

- 变量未赋值，不能直接使用
- 变量只在它所属的范围内有效 变量属于它所在的那一对大括号
- 一行上可以定义多个变量，但是不建议

```
public class VariableDemo2 {
  public static void main(String[] args) {
    //定义一个变量
    int a = 10;
    System.out.println(a);
    //如下定义多个变量时，写在同一行（注释部分）不如分开写
    /*
    int aa, bb;
    aa = 10;
    bb = 20;
    System.out.println(aa);
    System.out.println(bb);
    */
    int aa = 10;
    int bb = 20;
    System.out.println(aa);
```

```
    System.out.println(bb);
  }
}
}
12345678910111213141516171819
```

Java中的数据类型

数据类型：Java是一种强类型语言，针对每一个数据都给出了明确的数据类型

数据类型分类：
A:基本数据类型
B:引用数据类型(类，接口，数组)

基本数据类型

整数	占用字节数
byte	1
short	2
int	4
long	8

浮点数	占用字节数
float	4
double	8

字符	占用字节数
char	2

布尔	占用字节数
boolean	1

注意事项
A:整数默认是int类型，浮点数默认是double
B:定义long类型的数据的时候，要加L或者l，建议加L
定义float类型的数据的时候，要加F或者f，建议加F

数据类型转换

数据类型可以转换通过：隐式转换和强制转换

隐式转换：
byte,short,char – int – long – float – double 优先级依次递增
其中boolean不参与这样的运算

```
public class ConversionDemo {
  public static void main(String[] args) {
    //定义两个int类型的变量
    int a = 10;
    int b = 20;
    System.out.println(a+b);
    //下面这种写法报错，隐式转换变int类型，不能用byte接收
    //byte cc = aa+bb;
    int cc = aa + bb;
    System.out.println(cc);
  }
}
```

强制转换：目标类型变量名= (目标类型)(被转换的数据);
注意：虽然可以做强制转换，但是不建议，因为强制转换可能会有数据丢失。

```
public class ConversionDemo2 {
  public static void main(String[] args) {
    //定义两个变量，一个int类型，一个byte类型
    int a = 10;
    byte b = 20;
    byte e = (byte)(a + b);
    System.out.println(e);
  }
}
```

Java之运算符

Java中常用的运算符有

- 算术运算符
- 自增自减运算符
- 赋值运算符
- 关系运算符
- 逻辑运算符
- 三元运算符

算术运算符

算数运算符：+, -, *, /, % （加，减，乘，除，求余）

Note: 整数相除只能得到整数，想要得到小数，就必须有浮点数参与运算

字符与字符串参与的加法运算

字符参与加法运算: 其实是拿该字符在计算机中存储所表示的数据值来运算的
例：
‘a’ 97
‘A’ 65
‘0’ 48
字符串参与加法运算：其实这里不是加法，而是字符串的拼接

```
public class OperatorDemo2 {
  public static void main(String[] args) {
    //定义两个变量，一个int类型，一个char类型
    int a = 10;
    char ch = 'a';
    System.out.println(a + ch);  //输出 107

    //字符串做加法
    System.out.println("hello"+"world"); //输出 helloworld
    System.out.println("hello"+10);    //输出 hello10
    System.out.println("hello"+10+20); //输出 hello1020
    System.out.println(10+20+"hello"); //输出 30hello
  }
}
```

自增自减运算符

自增自减运算符：++, –

Note: ++和–可以放在变量的前面，也可以放在变量的后面。
单独使用一个变量的时候，放在变量的前面或者后面，效果一样。
参与其他操作的时候：
++在变量的后面： 先把变量做操作然后变量再++
++在变量的前面，先变量++，然后再操作

```
public class OperatorDemo {
  public static void main(String[] args) {
    //定义一个int类型的变量
    int a = 10;
    System.out.println("a:"+a); //输出：a:10
    int b = a++;                //输出：a:11 b:10
    //int b = ++a;              //输出：a:11 b:11
    System.out.println("a:"+a);
    System.out.println("b:"+b);
  }
}
```

赋值运算符

赋值运算符：

- 基本的赋值运算符：=
- 扩展的赋值运算符：+=,-=,...

a = a + 20 与 a += 20 有区别
例：
short s = 1;
s = s + 1;
// s参与运算会变int，将结果赋值给short s, 就会报错

扩展的赋值运算符隐含了强制类型转换
a += 20； 等价于 a =(a的数据类型)(a+20)；

```
public class OperatorDemo {
  public static void main(String[] args) {
    short s = 1;
    s += 1;
    System.out.println("s:"+s); // 输出 s:2
  }
}
```

关系运算符

关系运算符：==, !=, > ,>=, <, <=
关系运算符操作完毕的结果是boolean类型。

Note: 千万不要把==写成了=

逻辑运算符

逻辑运算符：

- 逻辑与 &&: 有false则false
- 逻辑或 ||: 有true则true
- 逻辑非 !: true则false,false则true

三元运算符

三元运算符：关系表达式?表达式1:表达式2;

执行流程：
A:计算关系表达式的值，看结果是true还是false
B:如果是true，表达式1就是结果
如果是false，表达式2就是结果

```
public class OperatorDemo {
  public static void main(String[] args) {
    //定义两个变量
    int a = 10;
    int b = 20;

    int c = (a>b)?a:b;
    System.out.println("c:"+c);
  }
}
```

比较两个整数是否相同

```
public class OperatorTest {
  public static void main(String[] args) {
    //定义两个int类型的变量
    int a = 10;
    int b = 10;
    boolean flag = (a==b)?true:false;
    System.out.println(flag); //输出 true
  }
}
```

Java之键盘录入

如何实现键盘录入数据呢? 目前使用JDK提供的类Scanner。
使用步骤：

- A: 导包
  import java.util.Scanner;

注意：在一个类中，有这样的顺序关系
package > import > class

- B: 创建键盘录入对象
  Scanner sc = new Scanner(System.in);
- C: 获取数据
  int i = sc.nextInt();

```
import java.util.Scanner;
public class ScannerDemo {
  public static void main(String[] args) {
    //创建键盘录入对象
    Scanner sc = new Scanner(System.in);

    //给出提示
    System.out.println("请输入一个整数：");
    //获取数据
    int i = sc.nextInt();

    //把获取的数据输出
    System.out.println("i:"+i);
    // 输出：  请输入一个整数:
    // 手动输出整数：99
    // i:99
  }
}
12345678910111213141516171 8
```

Java之选择语句

If语句

格式1：
if(关系表达式) {
语句体1;
}

格式2：
if(关系表达式) {
语句体1;
}else {
语句体2;
}
格式3：
if(关系表达式) {

语句体1;
}else if {
语句体2;
}
...
else{
语句体n+1;
}

程序中的数据测试注意事项

写程序的时候，做数据测试，应该测试这样的几种情况

- 正确数据
- 边界数据
- 错误数据

```
import java.util.Scanner;
public class IfTest2 {
  public static void main(String[] args) {
    //创建键盘录入对象
    Scanner sc = new Scanner(System.in);

    //给个提示
    System.out.println("请输入学生的考试成绩：");
    int score = sc.nextInt();
    //加入非法数据测试
    if(score>100 || score<0) {
      System.out.println("输入的数据有误");
    }else if(score>=90 && score<=100) {
      System.out.println("优秀");
    }else if(score>=80 && score<90) {
      System.out.println("好");
    }else if(score>=70 && score<80) {
      System.out.println("良");
    }else if(score>=60 && score<70) {
      System.out.println("及格");
    }else {
      System.out.println("不及格");
    }
  }
}
```

switch语句

switch语句格试：
switch(表达式) {
case值1:
语句体1;
break;
case值2:
语句体2;
break;
case值3:
语句体3;
break;
...
default:
语句体n+1;
break;
}

格式解释：
表达式：byte,short,int,char
JDK5以后可以是枚举,JDK7以后可以是字符串
case后面的值：用来和表达式的值进行匹配的
break：表示中断的意思
default：所有的值都和表达式不匹配，就执行default对应的内容

```
import java.util.Scanner;
public class SwitchDemo {
  public static void main(String[] args) {
    // 键盘录入数据
    Scanner sc = new Scanner(System.in);

    // 输出提示
    System.out.println("请输入一个整数(1-7)：");
    int weekDay = sc.nextInt();

    switch (weekDay) {
    case 1:
      System.out.println("星期一");
      break;
    case 2:
      System.out.println("星期二");
      break;
    case 3:
      System.out.println("星期三");
      break;
    case 4:
```

```
      System.out.println("星期四");
      break;
    case 5:
      System.out.println("星期五");
      break;
    case 6:
      System.out.println("星期六");
      break;
    case 7:
      System.out.println("星期日");
      break;
    default:
      System.out.println("输入数据有误");
      break;
    }
  }
}
```

Java之循环语句

for 循环

for循环语句的格式：
for (初始化语句; 判断条件语句; 控制条件语句) {
循环语句;
}

执行流程

1. 执行初始化语句
2. 执行判断条件语句，看其结果，是true还是false 。如果false，就结束循环 。如果true，就继续执行
3. 执行循环体语句
4. 执行控制条件语句
5. 回到B继续

for循环例子：打印水仙花数

水仙花数是什么?
一个三位数，各位数字的立方和等于该数本身
举例：153是一个水仙花数
111 + 555 + 333 = 1 + 125 + 27 = 153

```
public class ForTest4 {
```

```
  public static void main(String[] args) {
    //三位数的范围用for循环实现
    for(int x=100; x<1000; x++) {
      //获取各个位数
      int ge = x%10;
      int shi = x/10%10;
      int bai = x/10/10%10;

      //比较大小
      if((ge*ge*ge+shi*shi*shi+bai*bai*bai) == x) {
        //相等即是水仙花数
        System.out.println(x);
      }
    }
  }
}
```

while 循环

while循环语句的格式：
while(判断条件语句) {
循环语句;
}

完整格式：
初始化语句;
while(判断条件语句) {
循环体语句;
控制条件语句;
}

比较for循环与while循环

```
public class WhileDemo {
  public static void main(String[] args) {
    //控制台输出5次HelloWorld
    //for循环实现
    /*
    for(int x=1; x<=5; x++) {
      System.out.println("HelloWorld");
    }
    System.out.println("--------------");
    */
```

```
    //while循环实现
    int x=1;
    while(x<=5) {
      System.out.println("HelloWorld");
      x++;
    }
  }
}
```

Do while 循环

do...while循环的基本格式
do {
循环体语句;
}while(判断条件语句);

do...while 循环demo：

```
public class DoWhileDemo {
  public static void main(String[] args) {
    //在控制台输出5次hello world
    /*
    for(int x=1; x<=5; x++) {
      System.out.println("HelloWorld");
    }
    */

    int x=1;
    do {
      System.out.println("A");
      x++;
    }while(x<=5); //output: A A A A A

  }
}
12345678910111213141516 17
```

三种循环的区别

三种循环语句可以完成相同的事情，但是也有小区别
do...while循环语句至少执行一次循环体
而for和while会先判断再决定是否执行循环体

for循环和while循环的小区别
for循环结束后，初始化变量不能被使用
while循环结束后，初始化变量依然能被使用

推荐使用的顺序
（优先）for –（再）while – do...while

```
public class DoWhileDemo2 {
  public static void main(String[] args) {
    int x=1;
    while(x <= 5) {
      System.out.println("1");
      x++;
    }
    System.out.println(x); //output: 1 1 1 1 1 6
  }
}
12345678910
```

Java之循环嵌套

循环嵌套：就是循环体语句体本身是一个循环语句。

外循环控制的是行，内循环控制的是列。
Demo 1：输出一个4行5列的星星图案

```
public class ForForDemo {
  public static void main(String[] args) {
    for(int y=1; y<=4; y++) {
      for(int x=1; x<=5; x++) {
        System.out.print("*");
      }
      System.out.println();
    }
  }
}
```

Demo 2: 打印九九乘法表

```
public class ForForTest2 {
  public static void main(String[] args) {
    /*
    1*1=1
    1*2=2 2*2=4
```

```
    1*3=3 2*3=6 3*3=9
    ...
    */
    for(int x=1; x<=9; x++) {
      for(int y=1; y<=x; y++) {
        // \t:转移字符：表示一个tab键的位置
        System.out.print(y+"*"+x+"="+y*x+"\t");
      }
      System.out.println();
    }
  }
}
```

Java之跳转控制语句

跳转控制语句 break

break:中断
使用场景：

- A:switch语句中用于结束switch语句
- B:循环语句中，用于结束循环

```
public class BreakDemo {
  public static void main(String[] args) {
    //break可以用于结束当前的循环
    for(int x=1; x<=5; x++) {
      if(x == 3) {
        break;
      }
      System.out.println("HelloWorld");
    }
    System.out.println("-----------------------");

    //如果是多层循环，break结束的是离它最近的循环
    //如果要跳出外层循环
    //带标签的语句：
    //格式：标签名：语句
    wc:for(int x=1; x<=3; x++) {
      nc:for(int y=1; y<=4; y++) {
        if(y == 2) {
          break wc;
        }
        System.out.print("*");
      }
```

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

跳转控制语句 continue

continue: 继续
break和continue的区别

- break:跳出整个循环
- continue:跳出这一次的操作，进入下一次的执行

```
public class ContinueDemo {
  public static void main(String[] args) {
    for(int x=1; x<=5; x++) {
      if(x == 3) {
        continue;
      }
      System.out.println(x); //output: 1 2 4 5
    }
  }
}
12345678910
```

Java 之 Random

Random:用于产生随机数的类。用法和Scanner类似。
使用步骤：

- A:导包 import java.util.Random;
- B:创建对象 Random r = new Random();
- C:获取随机数 int number = r.nextInt(10);
  获取的范围：[0,10)，包括0不包括10

综合练习题：猜数字

```
import java.util.Random;

public class RandomTest {
  public static void main(String[] args) {
    //产生随机数
    Random r = new Random();
    //获取随机数
```

```
    int number = r.nextInt(100)+1; //随机数产生范围为【0, 100）

    while(true) {
      //键盘录入
      Scanner sc = new Scanner(System.in);
      //输出提示
      System.out.println("请输入你要猜的数据(1-100)：");
      int guessNumber = sc.nextInt();

      //比较数据
      if(guessNumber > number) {
        System.out.println("你猜的数据"+guessNumber+"大了");
      }else if(guessNumber < number) {
        System.out.println("你猜的数据"+guessNumber+"小了");
      }else {
        System.out.println("恭喜你，猜中了");
        break; //跳出循环
      }
    }
  }
}
```

Java之数组

数组：存储同一种类型的多个元素的容器

定义格式：
A:数据类型[] 数组名;(推荐使用的方式)
B:数据类型数组名[];

举例：

- int[] arr; 定义了一个int类型的数组，数组名是arr
- int arr[]; 定义了一个int类型的变量，变量名是arr数组

数组的初始化：所谓的初始化，其实就是为数组开辟内存空间，并为数据中的每个元素赋予初始值

数组初始化的两种方式：

- 动态初始化 只给出长度，由系统给出初始化值
- 静态初始化 只给出初始化值，由系统决定长度

动态初始化：
数据类型[] 数组名 = new 数据类型[数组长度];
int[] arr = new int[3];

静态初始化：
数据类型[] 数组名 = new 数据类型[]{元素1,元素2,元素3,...};
int[] arr = new int[]{1,2,3};

简化的格式：
数据类型[] 数组名 = {元素1,元素2,元素3,...};
简化格式的代码：
int[] arr = {1,2,3};

java中的内存分配

栈：存储的是局部变量
堆：存储的是new出来的东西 （东西：实体，对象）
方法区：面对对象部分
本地方法区：和系统相关
寄存器：给CPU使用

栈：

- 存储的是局部变量
  局部变量：定义在方法中的变量
- 使用完毕，立即回收

堆：

- 每一个对象都有地址值
- 每一个对象都有默认值
  byte, short, int, long – 0
  float, double – 0.0
  char – '\u0000'
  boolean – false
  引用类型 – null
- 数据使用完毕后，会在垃圾回收器空闲的时候被回收

int[] arr = new int[3];
System.out.println(arr)； output：001

数组操作的两个常见问题

问题一：
ArrayIndexOutOfBoundsException：数组索引越界异常
原因：访问了不存在的索引

问题二：
NullPointerException: 空指针异常
原因：null是指不再指向堆内存的数据，而我们还在访问
arr索引地址被null替代，无法访问原来的数组

```
public class ArrayDemo {
  public static void main(String[] args) {
    //定义数组
    int[] arr = {1,2,3};

    //访问数组中的元素
    //System.out.println(arr[3]);

    //引用数据类型：类，接口，数组
    //常量：null，可以赋值给引用数据类型，表示该引用不再指向堆内存的数据
    arr = null;
    System.out.println(arr[1]);
  }
}
```

数组常见操作

数组遍历

Note: 可以用arr.length得到数组长度

```
public class ArrayOperatorDemo {
  public static void main(String[] args) {
    //定义数组
    int[] arr = {11,22,33,44,55};
    //解决数数组中的元素个数问题，数组提供属性：length
    //用于获取数组中的元素个数
    //使用格式：数组名.length
    System.out.println("arr数组共有"+arr.length+"个元素");
    System.out.println("------------------");

    //标准写法
    for(int x=0; x<arr.length; x++) {
      System.out.println(arr[x]);
    }

  }
}
12345678910111213141516 17
```

数组获取最值

数组获取最值的思路：从该数组中获取参照物

1. 拿数组中的第一个元素做参照物
2. 遍历数组，从第二个元素开始，依次和参照物比较。如果元素比参照物大，就留下来做参照物
3. 整个比较完毕后，参照物就是最大的数据了

```
public class ArrayOperatorDemo2 {
  public static void main(String[] args) {
    //定义数组
    int[] arr = {12,45,98,73,60};

    //定义参照物
    int max = arr[0];
    for(int x=1; x<arr.length; x++) { //从第二个元素开始遍历
      if(arr[x] > max) {
        //如果比max大，替换原本的值
        max = arr[x];
      }
    }
    System.out.println("max:"+max);
    System.out.println("----------------------");

    //寻找最小值
    int min = arr[0];
    for(int x=1; x<arr.length; x++) {
      if(arr[x] < min) {
        min = arr[x];
      }
    }
    System.out.println("min:"+min);
  }
}
```