Java 入门- 预科班内容正课会重讲

内参保密不得外传

达内 Java 培优方向

陈子枢 主讲

今日头条极速版

2020

1 第三天：数据结构：基本类型+包装类型+数组+对象

1.1 数据结构

1.1.1 什么是数据结构?

圣贤说过：算法+数据结构=程序。

算法就是计算方法，那计算什么呢？是不需要数据？数据放哪呢？就放在对应的数据结构。

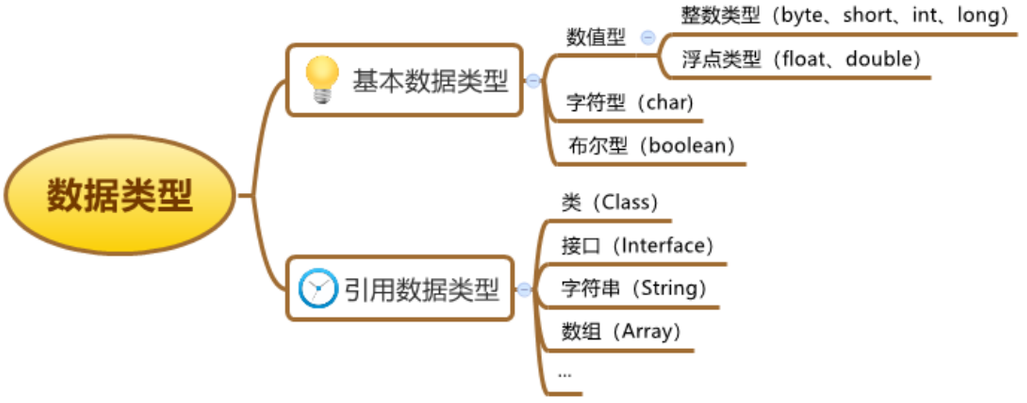
如整形放在 int 里，长整型放在 long 里，小数放在 double 里，字符串放在 String 里，一组数据就放数组里，当然还有更复杂的数据就放在对象 Object 里。

1.1.2 数据类型

Java 语言是一种强类型语言。第一、所有的变量必须先声明、后使用。第二、指定类型的变量只能接受和声明的类型匹配的类型值。强类型语言的好处是在编译阶段就可以发现源代码的错误，从而保证程序更加健壮。但也有缺点，有时程序运行中难以确定其类型，面对这种业务场景 java 的强类型就有些死板，索性后期 java 又推出了泛型弥补这个缺失。两者配合实现动静皆宜。不论如何，强类型语言的优点还是大于劣势的。

又出来一个新名词，泛型？什么是泛型大家先不用着急，后期专门有章节进行学习。

上面的话太官方，学过的同学好理解，没学过的同学不知所云。那我们就通俗点说，在 Java 中存储的数据都是有类型的，而且必须在编译时就确定其类型。Java 中有两类数据类型：



在 Java 的领域里，基本数据类型变量存的是数据本身，而引用类型变量存的是保存数据的空间地址。说白了，基本数据类型变量里存储的是直接放在抽屉里的东西，而引用数据类型变量里存储的是这个抽屉的钥匙，钥匙和抽屉一一对应。

1.2 基本类型

1.2.1 八种基本类型

基本类型（primitive type）这八种厉害了，内置的，直接使用。

	类型名称	字节空间	默认值	取值范围
整数型	byte	1	0	-2^7 到 2^7-1 或者 -128 到 127
	short	2	0	-2^{15} 到 $2^{15}-1$
	int	4	0	-2^{31} 到 $2^{31}-1$
	long	8	0L	-2^{63} 到 $2^{63}-1$
浮点型	float	4	0.0f	单精度，对小数部分的精度要求不高
	double	8	0.0d	双精度，精确的小数部分并操作值很大时
字符型	char	2	空格	0 到 65535
布尔型	boolean	1	false	真 true 假 false

注意：声明变量类型时，会给变量分配内存，按类型的所占空间数来分配，那这里就很讲究了，例如：如果年龄应该使用什么类型呢？最佳是 byte，0~100，能活 100 岁很了不起，非常高寿了。当然也有习惯的因素，开发者习惯 int age 来定义。例如：男女就两个值，习惯选择 boolean gender = false。false 代表女，true 就代表男。

所以记住一点，实际开发尽量用字节空间少的，内存是宝贵的稀缺的就那么点，少占点。

1.2.2 默认初始值

八种基本类型的初始值:

```
package javase.base;

import org.junit.Test;

//打印每个基本类型的初始值
public class TestInitValue {
    //八种基本类型初始值
    //声明成局部变量必须设置初始化值，所以使用成员变量
    boolean b;
    char c;
    byte bt;
    short s;
    int i;
    long l;
    float f;
    double d;
    Integer r1;

    @Test
    public void test() {
        System.out.println("布尔类型: "+b);
        System.out.println("字符类型: " + (int)c);
        System.out.println("字节类型: " + bt);
        System.out.println("短整形: " + s);
        System.out.println("整形: " + i);
        System.out.println("长整形: " + l);
        System.out.println("单精度浮点类型: " + f);
        System.out.println("双精度浮点类型: " + d);
        System.out.println("包装类型: " + r1);
    }
}
```

执行结果:

```
布尔类型: false
字符类型: 0
字节类型: 0
短整形: 0
整形: 0
长整形: 0
单精度浮点类型: 0.0
```

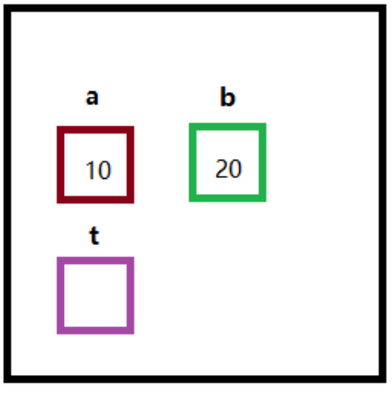
```
双精度浮点类型：0.0
包装类型：null
```

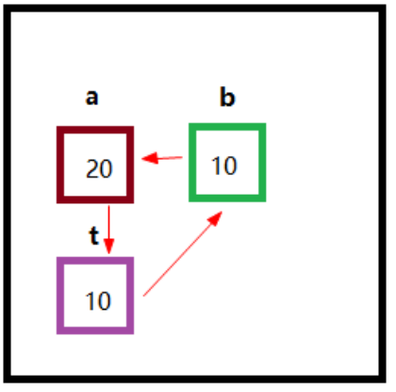
1.2.3 交换值

```
package javase.basetype;

import org.junit.Test;

public class TestSwap {
    @Test
    public void swap() {
        int a = 10;
        int b = 20;
        System.out.println("交换前：a="+a+",b="+b);

        int t = b;     //中间的临时变量，用完就抛弃，过河就拆桥
        b = a;
        a = t;

        System.out.println("交换后：a="+a+", b="+b);
    }
}
```

当基本数据类型作为参数进行传递时，传递的是值，当有一个方法中的值发生改变，则对另一个方法中的值没有任何影响(各自是独立的)

交换过程：





1.3 包装类型

1.3.1 包装类

Wrapper Class 包装类：

序号	类型名称	基本类型	包装类型	父类
1	字节型	byte	Byte	Number
2	短整型	short	Short	
3	整型	int	Integer	
4	长整型	long	Long	
5	单精度浮点型	float	Float	
6	双精度浮点型	double	Double	
7	字符型	char	Character	Object
8	布尔型	boolean	Boolean	

可以看出 java 是两种类型相结合的产物，各有特长，那实际开发中如何使用呢？大家可以先简单记忆一下它的规则，日后我们开发中都遵循这个原则：

- 在方法体内部尽量使用基本类型，占空间小，用完就释放内存，节约内存
- 在 pojo 时尽量使用包装类型，例如 mybatis 持久层框架就必须使用包装类型，否则可能出错哦。（先记忆下来，后期讲到 mybatis 框架，大家可以验证）

1.3.2 最大值最小值

```
package javase.basetype;

import org.junit.Test;

public class TestValueLimit {
    @Test
    public void testNumber() {
        System.out.println("\n==整形值类型==================");

        byte minByte = Byte.MIN_VALUE;
        byte maxByte = Byte.MAX_VALUE;
        System.out.println("byte 取值范围："+ minByte + "~"+maxByte);

        short minShort = Short.MIN_VALUE;
        short maxShort = Short.MAX_VALUE;
        System.out.println("short 取值范围："+ minShort + "~"+maxShort);
```

```
            int minInteger = Integer.MIN_VALUE;
            int maxInteger = Integer.MAX_VALUE;
            System.out.println("int 取值范围: "+ minInteger +
"~"+maxInteger);

            long minLong = Long.MIN_VALUE;
            long maxLong = Long.MAX_VALUE;
            System.out.println("long 取值范围: "+ minLong + "~"+maxLong);
        }

        @Test
        public void testFloat() {
            System.out.println("\n==浮点值类型==================");

            float minFloat = Float.MIN_VALUE;
            float maxFloat = Float.MAX_VALUE;
            System.out.println("float 取值范围: "+ minFloat + "~"+maxFloat);

            double minDouble = Double.MIN_VALUE;
            double maxDouble = Double.MAX_VALUE;
            System.out.println("double 取值范围: "+ minDouble +
"~"+maxDouble);
        }
    }
```

执行结果:

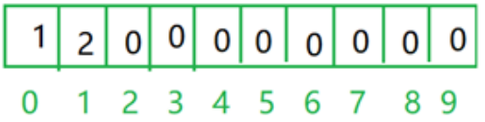
```
==整形值类型==================
byte 取值范围: -128~127
short 取值范围: -32768~32767
int 取值范围: -2147483648~2147483647
long 取值范围: -9223372036854775808~9223372036854775807

==浮点值类型==================
float 取值范围: 1.4E-45~3.4028235E38
double 取值范围: 4.9E-324~1.7976931348623157E308
```

1.4 数组

1.4.1 创建数组

我们前面学的基本变量只能存储一个值，那要存很多值呢？java 有很多方式，那我要学个最简单的有吗？当然有，那就是马上要介绍的数组。

```
int[] arr = new int[10];
arr[0] = 1;
arr[1] = 2;
```




```
package javase.base;

public class TestArea {
    public static void main(String[] args) {
        //声明数组，数组有 10 个元素
        int[] scores = new int[10];
        //int scores[] = new int[10];    //也可以这样定义

        //对数组进行初始化
        for(int i=0;i<10;i++) {
            scores[i] = i*10;
        }

        //遍历数组打印数组的每个元素值
        for(int i=1;i<11;i++) {
            System.out.println(scores[i-1]);
        }

        System.out.println();

        //反向遍历
        for(int i=scores.length-1; i>=0; i--) {
            System.out.println(scores[i]);
        }
    }
}
```

注意：数组的下标是从零开始，第一个元素为 x[0]，最后一个元素为 x[length-1]。如果访问 x[length]就会报错，数组越界。所以访问数组元素时，一定要注意下标的范围。

1.4.2 直接初始化

```
        int[] scores = {1,2,3,4,5};

        //遍历数组打印数组的每个元素值
        for(int i=0; i<scores.length; i++) {
            System.out.println(scores[i]);
        }
```

上面是习惯的简写方式，正规写法：

```
int[] scores = new int[]{1,2,3,4,5};
```

1.4.3 异常：数组索引越界

数组很娇气特别爱报错，所以使用时一定要思考清晰。

数组只有 5 个元素，却去访问第 10 个元素

```
int[] scores = new int[]{1,2,3,4,5};
System.out.println(scores[10]);
```

执行结果：

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
    at javase.base.TestArea.main(TestArea.java:16)
```

注意这种错误很隐秘，编译时不报错，运行时才会报错，所以开发时一定要注意其取值。

TestArea.java

自己处理异常：

```
package javase.base;

public class TestArea {
    public static void main(String[] args) {
        try {
            int[] scores = new int[]{1,2,3,4,5};
            System.out.println(scores[10]);
        }catch(Exception e) {
            System.out.println("异常的类："+e.getClass());
            System.out.println("异常信息："+e.getMessage());
        }
    }
}
```

执行结果：

```
异常的类：class java.lang.ArrayIndexOutOfBoundsException
异常信息：10
```

1.5 数组案例

1.5.1 随机数组

有时我们需要些试验数据，都自己去写那不是累死了，写 5 个很轻松，写 500 个呢？

Java 提供了 api 函数(java 习惯叫方法，c 习惯叫函数，其实只是个名称作用都一样)，可以轻松产生随机数，然后赋值给数组即可。

```
package javase.base;
```

```
import java.util.Random;      //导包，用谁引入谁

public class TestArea {

    public static void main(String[] args) {
        int[] arr = new int[5];
        for(int i = 0; i<arr.length; i++ ) {
            //产生 10 以内的随机数
            arr[i] = new Random().nextInt(10);
        }

        for(int i=0; i<arr.length; i++) {
            System.out.println(arr[i]);
        }
    }

}
```

nextInt(n)该方法的作用是生成一个随机的 int 值，该值介于[0,n)的区间，也就是 0 到 n 之间的随机 int 值，包含 0 而不包含 n。

多执行几次，可以看到每次执行的结果都不同，随机产生。

1.5.2 考试成绩统计

```
package javase.base;

import java.util.Arrays;

public class TestScore {
    public static void main(String[] args) {
        //语文的五次考试成绩
        int[] scores = {100,70,80,60,88};

        //1. 合计值
        int sum = 0;
        for(int i=0;i<scores.length;i++) {
            sum += scores[i];
        }
        System.out.println("总成绩: " + sum);

        //2. 平均值
        int avg = sum/scores.length;
        System.out.println("平均成绩: " + avg);

        Arrays.sort(scores);  //排序，升序

        //3. 最大值
        int max = scores[scores.length-1];
```

```
            System.out.println("历史最高分: " + max);

            //4. 最小值
            int min = scores[0];
            System.out.println("历史最低分: " + min);
        }
    }
```

1.5.3 二维数组案例

```
    package hello;

    public class TestArea {
        /*
         *  公司年销售额求和
            某公司按照季度和月份统计的数据如下:
            第一季度: 22, 66, 44
            第二季度: 77, 33, 88
            第三季度: 25, 45, 65
            第四季度: 11, 66, 99
            分析: 将题目中的数据用二位数组来表示
            int[][] arr ={{22,66,44},{77,33,88},{25,45,65},{11,66,99}}
         */
         public static void main(String[] args) {
                int[][] arr
={{22,66,44},{77,33,88},{25,45,65},{11,66,99}};
                printArray(arr);
            }
            public static void printArray(int[][] arr){
                int sum=0;
                for(int x=0;x<arr.length;x++){
                    for(int y=0;y<arr[x].length;y++){
                        sum+=arr[x][y];
                    }
                }
                System.out.println(sum);
            }
    }
```

1.6 对象

1.6.1 需求

课程：course
1. 课程编号：id
2. 课程名称：name

老师：teacher
1. 编号：id
2. 姓名：name
3. 性别：sex
4. 生日：birthday
5. 职称：prof
6. 系：dpart

学生：
1. 编号：id
2. 姓名：name
3. 性别：sex
4. 生日：birthday
5. 班级：className

1.6.2 课程

```
package cn.teud.object;

public class Course {
    private Integer id;
    private String name;

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    @Override
```

```
    public String toString() {
        return "Course [id=" + id + ", name=" + name + "]";
    }
}
```

1.6.3 老师

```
package cn.teud.object;

import java.util.Date;

public class Teacher {
    private Integer id;
    private String name;
    private String sex;
    private Date birthday;
    private String prof;
    private String dpart;

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getSex() {
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
    }
    public Date getBirthday() {
        return birthday;
    }
    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }
    public String getProf() {
        return prof;
    }
    public void setProf(String prof) {
        this.prof = prof;
    }
    public String getDpart() {
        return dpart;
    }
    public void setDpart(String dpart) {
```

```
            this.dpart = dpart;
        }
        @Override
        public String toString() {
            return "Teacher [id=" + id + ", name=" + name + ", sex=" + sex
+ ", birthday=" + birthday + ", prof=" + prof
                    + ", dpart=" + dpart + "]";
        }
    }
```

1.6.4 学生

```
    package cn.teud.object;

    import java.util.Date;

    public class Student {
        private Integer id;
        private String name;
        private String sex;
        private Date birthday;
        private String className;

        public Integer getId() {
            return id;
        }
        public void setId(Integer id) {
            this.id = id;
        }
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }
        public String getSex() {
            return sex;
        }
        public void setSex(String sex) {
            this.sex = sex;
        }
        public Date getBirthday() {
            return birthday;
        }
        public void setBirthday(Date birthday) {
            this.birthday = birthday;
        }
        public String getClassName() {
            return className;
        }
        public void setClassName(String className) {
            this.className = className;
        }
```

```
        @Override
        public String toString() {
            return "Student [id=" + id + ", name=" + name + ", sex=" + sex
+ ", birthday=" + birthday + ", className="
                    + className + "]";
        }
    }
```

1.6.5 访问

```
    package cn.teud.object;

    import java.util.Date;

    import org.junit.Test;

    public class TestObj {
        @Test
        public void course() {
            Course course = new Course();
            course.setId(1);
            course.setName("历史");

            System.out.println("==课程信息==");
            System.out.println( "课程编号: " + course.getId() );
            System.out.println( "课程名称: " + course.getName() );

            System.out.println("测试: "+course);
            System.out.println();
        }

        @Test
        public void teacher() {
            Teacher t = new Teacher();
            t.setId(1);
            t.setName("李教授");
            t.setSex("男");
            t.setBirthday(new Date());
            t.setProf("副教授");
            t.setDpart("中文系");

            System.out.println("==老师信息==");
            System.out.println( "编号: " + t.getId() );
            System.out.println( "姓名: " + t.getName() );
            System.out.println( "性别: " + t.getSex() );
            System.out.println( "生日: " + t.getBirthday() );
```

```
        System.out.println( "职称: " + t.getProf() );
        System.out.println( "系: " + t.getDpart() );

        System.out.println("测试: "+t);
        System.out.println();
    }

    @Test
    public void student() {
        Student stu = new Student();
        stu.setId(7);
        stu.setName("詹姆士邦德");
        stu.setSex("男");
        stu.setBirthday(new Date());
        stu.setClassName("军情六处");

        System.out.println("==学生信息==");
        System.out.println( "编号: " + stu.getId() );
        System.out.println( "姓名: " + stu.getName() );
        System.out.println( "性别: " + stu.getSex() );
        System.out.println( "生日: " + stu.getBirthday() );
        System.out.println( "班级: " + stu.getClassName() );

        System.out.println("测试: "+stu);
        System.out.println();
    }
}
```

1.7 小结

1.7.1 数据结构的作用?

数据结构就是来存储数据的,选择业务最佳的数据结构,第一、保证数据可以保存下来,这样计算机才能进行加工处理。第二、选择合适的数据结构,做到存储空间和计算性能上的平衡。

1.7.2 基本类型和包装类型的差异?

基本类型占用的内存空间小,计算性能快。一般在方法内部,局域变量均采用基本类型。方法的参数尽量采用基本类型。

包装类型拥有更多底层提供的方法，我们无需自己处理，使用起来更加方便。但它占用的内存空间较大，性能比基本类型低。通常在对象的成员变量时使用。

1.7.3 基本类型、数组、对象之间差异?

基本类型只能存放单值;

数组可以存放一组一个类型的数据;

对象是万能，万物的复杂结构都可用用对象表示；所有在 java 中万物皆对象!