第 6 天 Java 基础语法

今日内容介绍

- 自定义类
- ArrayList 集合

第1章 引用数据类型（类）

1.1 引用数据类型分类

提到引用数据类型（类），其实我们对它并不陌生，如使用过的 Scanner 类、Random 类。
我们可以把类的类型为两种：

- 第一种，Java 为我们提供好的类，如 Scanner 类，Random 类等，这些已存在的类中包含了很多的方法与属性，可供我们使用。
- 第二种，我们自己创建的类，按照类的定义标准，可以在类中包含多个方法与属性，来供我们使用。

这里我们主要介绍第二种情况的简单使用。

1.2 自定义数据类型概述

我们在 Java 中，将现实生活中的事物抽象成了代码。这时，我们可以使用自定义的数据类型（类）来描述（映射）现实生活中的事物。

类，它是引用数据类型，与之前学习的所有引用数据类型相同，自定义类也是一种数据类型。只是自定义类型并非 Java 为我们预先提供好的类型，而是我们自己定义的一种引用数据类型用来描述一个事物。

1.3 类的定义与使用

java 代码映射成现实事物的过程就是定义类的过程。

我们就拿一部手机进行分析，它能用来做什么呢？它可以打电话，上网，聊微信等，这些就是手机所提供的功能，也就是方法；手机也有它的特征，如颜色、尺寸大小、品牌型号等，这些就是手机的特征，也就是属性。

目前，我们只关注类中的属性，类中的方法在面向对象部分再进行学习。

1.3.1 类的定义格式

- 类的定义格式

```
创建 java 文件，与类名相同
public class 类名{
    数据类型  属性名称 1;
    数据类型  属性名称 2;
    …
}
```

通过类的定义格式，来进行手机类的描述，如下所示

```
public class Phone {
    /*
     * 属性
     */
    String brand;// 品牌型号
    String color;// 颜色
    double size; // 尺寸大小
}
```

上述代码，就是创建一个类的的过程，类的名称我们给起名为 Phone，类中包含了三个属性（brand 品牌型号、color 颜色、size 尺寸大小）。注意，类中定义的属性没有个数要求。

1.3.2 类的使用格式

Phone 类定义好后，我们就可以使用这个类了，使用方式和使用引用数据类型 Scanner 类相似。格式如下：

```
导包：我们将所有的类放到同一个文件夹下，可以避免导包。
创建对象：数据类型  变量名 = new 数据类型();
调用方法：目前我们定义的自定义类不涉及方法，只是属性（自定义类中的方法部分在面向对象部分讲解）
访问属性：变量名.属性 （这是当前的方式，后期会采取调用方法的方式替代掉直接访问的方式来完成对属性的访问。）
```

当有了 Phone 数据类型的变量后，我们就可以使用 Phone 类中的属性了。对属性的访问我们来演示一下，如下所示：

```
public class Test {
    public static void main(String[] args) {
        //定义了一个 Phone 类型的变量 p
        Phone p = new Phone();
        /*
         * 通过 p,使用 Phone 中的属性
         */
        //访问 p 中的 brand 品牌属性
        p.brand = "苹果 6s";
```

Comment [L1]: 为 p 中 brand 属性赋值为”苹果 6s”

```
        //访问 p 中的 color 颜色属性
        p.color = "白色";
        //访问 p 中的 size 尺寸大小属性
        p.size = 5.5;

        System.out.println("手机品牌为" + p.brand);
        System.out.println("手机颜色为" + p.color);
        System.out.println("手机尺寸大小为" + p.size);
    }
}
```

Comment [L2]: 为 p 中 color 属性赋值为"白色"

Comment [L3]: 为 p 中 size 属性赋值为 5.5

Comment [L4]: 使用 p 中 brand 属性值参与运算

Comment [L5]: 使用 p 中 color 属性值参与运算

Comment [L6]: 使用 p 中 size 属性值参与运算

运行结果如下所示

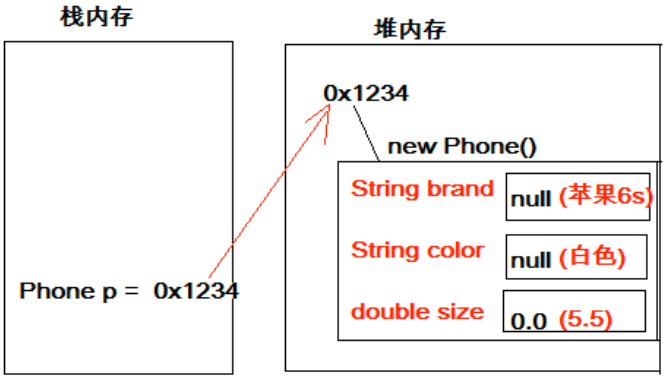
```
D:\java>java Test
手机品牌为苹果6s
手机颜色为白色
手机尺寸大小为5.5
```

图1-1　运行结果

1.3.3 自定义类型注意事项与内存图

上述代码中，通过类 Phone 创建出来的变量 p，它相当于我们生活中的盒子，里面包含了它能够使用的属性。

- 通过 p.属性名 就可以对属性进行操作
- 与引用类型数组类似，引用类型的自定义类型的变量，直接变量时，结果为对象地址值，这里可以通过内存图简单解释。



1.4 自定义类型练习

学习了引用数据类型（类）以后，我们就能够使用类描述任何东西了。看几个具体的描述，如下：

- 电饭锅，包含属性（品牌、容量大小、颜色等）
- 汽车，包含属性（品牌、排量、类型等）
- 学生，包含属性（姓名，年龄，性别等）

第2章 ArrayList 集合

在前面我们学习了数组，数组可以保存多个元素，但在某些情况下无法确定到底要保存多少个元素，此时数组将不再适用，因为数组的长度不可变。例如，要保存一个学校的学生，由于不停有新生来报道，同时也有学生毕业离开学校，这时学生的数目很难确定。为了保存这些数目不确定的元素，JDK 中提供了一系列特殊的类，这些类可以存储任意类型的元素，并且长度可变，统称为集合。在这里，我们先介绍 ArrayList 集合，其他集合在后续课程中学习。

ArrayList 集合是程序中最常见的一种集合，它属于引用数据类型（类）。在 ArrayList 内部封装了一个长度可变的数组，当存入的元素超过数组长度时，ArrayList 会在内存中分配一个更大的数组来存储这些元素，因此可以将 ArrayList 集合看作一个长度可变的数组。

2.1 集合的创建

创建集合的常用格式在此说明一下：

导包：import java.util.ArrayList;

创建对象：与其他普通的引用数据类型创建方式完全相同，但是要指定容器中存储的数据类型：

```
ArrayList<要存储元素的数据类型> 变量名 = new ArrayList<要存储元素的数据类型>();
```

- 集合中存储的元素，只能为<>括号中指定的数据类型元素；
- “<要存储元素的数据类型>”中的数据类型必须是引用数据类型，不能是基本数据类型；

下面给出 8 种基本数据类型所对应的引用数据类型表示形式：

基本数据类型	对应的引用数据类型表示形式
byte	Byte
short	Short
Int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

我们通过举几个例子，来明确集合的创建方式：

- 存储 String 类型的元素

```
ArrayList<String> list = new ArrayList<String>();
```

- 存储 int 类型的数据

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

- 存储 Phone 类型的数据

```
ArrayList<Phone> list = new ArrayList<Phone>();
```

2.2 集合中常用方法

接下来，我们来学习下 ArrayList 集合提供的一些常用方法，如下表：

方法声明	功能描述
boolean add（Object obj）	将指定元素 obj 追加到集合的末尾
Object get（int index）	返回集合中指定位置上的元素
int size（）	返回集合中的元素个数

通过代码演示上述方法的使用。ArrayListDemo01.java

```
import java.util.ArrayList;
public class ArrayListDemo01 {
    public static void main(String[] args) {
        // 创建 ArrayList 集合
        ArrayList<String> list = new ArrayList<String>();
        // 向集合中添加元素
        list.add("stu1");
        list.add("stu2");
        list.add("stu3");
        list.add("stu4");
        // 获取集合中元素的个数
        System.out.println("集合的长度: " + list.size());
        // 取出并打印指定位置的元素
        System.out.println("第 1 个元素是: " + list.get(0));
        System.out.println("第 2 个元素是: " + list.get(1));
        System.out.println("第 3 个元素是: " + list.get(2));
        System.out.println("第 4 个元素是: " + list.get(3));
    }
}
```

强调一点，ArrayList 集合相当于是一个长度可变的数组，所以访问集合中的元素也是采用索引方式访问，第一个元素存储在索引 0 的位置，第二个元素存储在索引 1 的位置，依次类推。

2.3 集合的遍历

通过集合遍历，得到集合中每个元素，这是集合中最常见的操作。集合的遍历与数组的遍历很像，都是通过索引的方式，集合遍历方式如下：ArrayListDemo02.java

```
1 import java.util.ArrayList;
2 public class ArrayListDemo02 {
3     public static void main(String[] args) {
4         //创建 ArrayList 集合
5         ArrayList<Integer> list = new ArrayList<Integer>();
6         //添加元素到集合
7         list.add(13);
8         list.add(15);
```

```
9           list.add(22);
10          list.add(29);
11          //遍历集合
12          for (int i = 0; i < list.size(); i++) {
13              //通过索引，获取到集合中每个元素
14              int n = list.get(i);
15              System.out.println(n);
16          }
17      }
18  }
```

Comment [L7]: 获取集合中元素的个数

Comment [L8]: 获取集合中指定位置上的元素值

上述代码中，第 5 行定义了一个可以存储 int 元素的集合；第 7-10 行，实现将 int 类型数值存储到集合中；第 12-16 行，实现遍历集合元素。这里要强调一点，get 方法返回值的类型为集合中元素的类型。

2.4 集合中的常用方法补充

ArrayList 集合提供的一些常用方法，如下表：

方法声明	功能描述
boolean add（int index,　Object obj）	将指定元素 obj 插入到集合中指定的位置
Object remve（int index）	从集合中删除指定 index 处的元素，返回该元素
void clear（）	清空集合中所有元素
Object set（int index, Object obj）	用指定元素 obj 替代集合中指定位置上的元素

- boolean add（int index,　Object obj）
  - 功能：在集合中指定 index 位置，添加新元素 obj
  - 功能说明：假设集合 list 中有元素[“java”,“javaEE”]，当使用 add(1,“javaWeb”)后，集合 list 中的元素为[“java”，“javaWeb”，“JavaEE”]。
- Object set（int index, Object obj）
  - 功能：用指定元素 obj 替代集合中指定 index 位置的元素
  - 功能说明：假设集合 list 中有元素[“java”,“javaEE”]，当使用 set(0,“javaWeb”)后，集合 list 中的元素为[“javaWeb”，“JavaEE”]。
- Object remve（int index）
  - 功能：从集合中删除指定 index 处的元素，返回该元素
  - 功能说明：假设集合 list 中有元素[“java”，“javaEE”]，当使用 remove(0)后，集合 list 中的元素为[“JavaEE”]，返回值为“java”。
- void clear（）
  - 功能：清空集合中所有元素
  - 功能说明：假设集合 list 中有元素[“java”，“javaEE”]，当使用 clear()后，集合 list 中的元素为空[]。

第3章 随机点名器案例

3.1 案例介绍

随机点名器，即在全班同学中随机的找出一名同学，打印这名同学的个人信息。

此案例在我们昨天课程学习中，已经介绍，现在我们要做的是对原有的案例进行升级，使用新的技术来实现。

我们来完成随机点名器，它具备以下 3 个内容：

- 存储所有同学姓名
- 总览全班同学姓名
- 随机点名其中一人，打印到控制台

3.2 案例需求分析

全班同学中随机的找出一名同学，打印这名同学的个人信息。

我们对本案例进行分析，得出如下分析结果：

1.存储全班同学信息（姓名、年龄）

2.打印全班同学每一个人的信息（姓名、年龄）

3.在班级总人数范围内，随机产生一个随机数，查找该随机数所对应的同学信息（姓名、年龄）

随机点名器明确地分为了三个功能。如果将多个独立功能的代码写到一起，则代码相对冗长，我们可以针对不同的功能可以将其封装到一个方法中，将完整独立的功能分离出来。

而在存储同学姓名时，如果对每一个同学都定义一个变量进行姓名存储，则会出现过多孤立的变量，很难一次性将全部数据持有。此时，我们采用 ArrayList 集合来解决多个学生信息的存储问题。

3.3 实现代码步骤

每名学生都拥有多项个人信息，为了方便管理每个人的信息，我们对学生信息进行封装，编写 Student.java 文件

```
/**
 * 学生信息类
 */
public class Student {
    String name; //姓名
    int age; //年龄
}
```

上述代码中，对学生信息（姓名、年龄）进行了封装。这样做的好处在于，以后只要找到这名学生，就能够知道他的每项个人信息了。

接下来我们编写 CallName.java 文件，完成程序的编写。

- main 方法中调用三个独立方法

```
    public static void main(String[] args) {
        ArrayList<Student> list = new ArrayList<Student>(); //1.1 创建一个可以存储多个同学名字的容器
        /*
         * 1.存储全班同学信息
         */
        addStudent(list);
        /*
         * 2.打印全班同学每一个人的信息（姓名、年龄）
         */
        printStudent(list);
        /*
         * 3.随机对学生点名，打印学生信息
         */
        randomStudent(list);
    }
```

- 存储所有学生的个人信息

```
    /**
     * 1.存储全班同学名字
     */
    public static void addStudent(ArrayList<Student> list) {
        //键盘输入多个同学名字存储到容器中
        Scanner sc = new Scanner(System.in);
        for (int i = 0; i < 3; i++) {
            //创建学生
            Student s = new Student();
            System.out.println("存储第"+i+"个学生姓名：");
            s.name = sc.next();
            System.out.println("存储第"+i+"个学生年龄：");
            s.age = sc.nextInt();
            //添加学生到集合
            list.add(s);
        }
    }
```

上述方法中，方法参数 list 中用来表示已存储所有学生。通过 Scanner，完成新学生信息（姓名，年龄）的录入，并将学生添加到集合中。

- 打印全班同学每一个人的信息

```
    /**
     * 2.打印全班同学每一个人的信息（姓名、年龄）
     */
    public static void printStudent (ArrayList<Student> list) {
        for (int i = 0; i < list.size(); i++) {
```

```
            Student s = list.get(i);
            System.out.println("姓名: "+s.name +",年龄: "+s.age);
        }
    }
```

上述方法中，方法参数 list 中用来表示已存储所有学生。通过遍历集合中的每个元素，得到每个同学信息，并输出打印。

- 随机对学生点名，打印学生信息

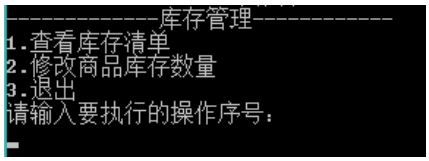
```
    /**
     * 3.随机对学生点名，打印学生信息
     */
    public static void randomStudent (ArrayList<Student> list) {
        //在班级总人数范围内，随机产生一个随机数
        int index = new Random().nextInt(list.size());
        //在容器（ArrayList 集合）中，查找该随机数所对应的同学信息（姓名、年龄）
        Student s = list.get(index);
        System.out.println("被随机点名的同学: "+s.name + ", 年龄:" + s.age);
    }
```

上述方法中，通过随机数类 Random 产生一个从 0 到集合长度的随机索引。使用该索引获取 ArrayList 集合中对应的值，便得到了全班同学的随机学生信息并打印。

第4章 库存管理案例

4.1 案例介绍

现在，我们将原有的库存管理案例，采用更好的集合方式实现。



将对下列功能进行方法封装：

- 打印库存清单功能
- 库存商品数量修改功能
- 退出程序功能

4.2 案例需求分析

管理员能够进行的操作有 3 项（查看、修改、退出），我们可以采用（switch）菜单的方式来完成。

----------库存管理----------

1.查看库存清单
2.修改商品库存数量
3.退出
请输入要执行的操作序号：

每一项功能操作，我们采用方法进行封装，这样，可使程序的可读性增强。

选择“1.查看库存清单”功能，则控制台打印库存清单

选择“2.修改商品库存数量”功能，则对每种商品库存数进行更新

选择“3.退出”功能，则退出库存管理，程序结束

4.3 实现代码步骤

每种库存商品都拥有多项商品信息，为了方便管理每种商品的信息，我们对商品信息进行封装，编写 Goods.java 文件

```
/*
 * 库存商品类
 */
public class Goods {
    String brand; // 商品品牌型号
    double size; // 商品尺寸大小
    double price; // 商品价格
    int count; // 商品库存个数
}
```

上述代码中，对商品信息（品牌、尺寸、价格、库存数）进行了封装。这样做的好处在于，以后只要找到这个商品，就能够知道该商品的每项信息了。

编写 Demo 库存管理.java，完成如下功能：

- 功能菜单

```
    /**
     * 库存管理功能菜单
     * @return 管理员键盘输入的功能操作序号
     */
    public static int chooseFunction() {
        System.out.println("-------------库存管理------------");
        System.out.println("1.查看库存清单");
        System.out.println("2.修改商品库存数量");
        System.out.println("3.退出");
        System.out.println("请输入要执行的操作序号：");
        //接收键盘输入的功能选项序号
        Scanner sc = new Scanner(System.in);
        int choose = sc.nextInt();
        return choose;
    }
```

上述方法用来完成库存管理功能菜单的显示、接收管理员选择的功能操作序号。这是完成了案例的第一步。接下来完成“查看、修改、退出”这三项功能。

- 编写 main 主方法，调用库存管理功能菜单方法，与“查看、修改、退出”这三个方法。

```
    public static void main(String[] args) {
        //记录库存商品信息
        ArrayList<Goods> list = new ArrayList<Goods>();
        //添加商品到库存
        addStore(list);

        //通过 while 循环模拟管理员进行功能重复选择操作
        while (true) {
            //打印功能菜单操作,接收键盘输入的功能选项序号
            int choose = chooseFunction();
            //执行序号对应的功能
            switch (choose) {
            case 1://查看库存清单
                printStore(list);
                break;
            case 2://修改商品库存数量
                update(list);
                break;
            case 3://退出
                exit();
                return;
            default:
                System.out.println("---------------------------------");
                System.out.println("功能选择有误，请输入正确的功能序号!");
                break;
            }
        }
    }
```

在主方法中，创建了 ArrayList 集合，用来存储库存商品信息，通过接收到的功能选项序号，进行 switch 语句判断后，调用对应的功能方法。

- 查看库存清单功能

```
    /**
     * 查看库存清单
     */
    public static void printStore(ArrayList<Goods> list) {
        //统计总库存个数、统计库存总金额
        int totalCount = 0;
        double totalMoney = 0.0;
        //列表顶部
        System.out.println("--------------------------  查 看 库 存 清 单
-------------------------");
        System.out.println("品牌型号      尺寸 价格 库存数");
        //列表中部
        for (int i = 0; i < list.size(); i++) {
```

```
                Goods item = list.get(i);
                System.out.println(item.brand+""+item.size+"   "+item.price+"  "+
item.count);
                //统计总库存个数、统计库存总金额
                totalCount += item.count;
                totalMoney += item.count * item.price;
            }
            //列表底部

    System.out.println("------------------------------------------------------------
-");
            System.out.println("总库存数："+totalCount);
            System.out.println("库存商品总金额："+totalMoney);
        }
```

上述方法用来完成打印库存清单功能，参数 list 表示库存商品相关信息集合。

- 修改商品库存数量功能

```
        /**
         * 修改商品库存数量
         */
        public static void update(ArrayList<Goods> list){
            System.out.println("------------修改商品库存数量-----------");
            for (int i = 0; i < list.size(); i++) {
                Goods item = list.get(i);
                System.out.println("请输入"+ item.brand +"商品库存数量");
                item.count = new Scanner(System.in).nextInt();
                list.set(i, item);
            }
        }
```

上述方法用来完成修改商品库存数量功能，参数 list 表示库存商品相关信息集合。

- 退出功能

```
        /**
         * 退出
         */
        public static void exit(){
            System.out.println("----------------退出---------------");
            System.out.println("您已退出系统");
        }
```

上述方法用来完成退出程序的功能

第5章 总结

5.1知识点总结

- 引用数据类型（类）
  - 类的类型为两种：
    - 第一种，Java 为我们提供好的类，如 Scanner 类，Scanner 类等，这些已存在的类中包含了很多的方法与属性，可供我们使用。
    - 第二种，我们自己创建的类，按照类的定义标准，可以在类中包含多个方法与属性，来供我们使用。
  - 创建类的格式

```
public class 类名 {
    //可以定义属性
    //也可以定义方法
}
```

  - 使用类的格式：

```
类名 变量名 = new 类名();
```

  - 使用类中的属性与方法格式

```
使用属性： 变量名.属性
使用方法： 变量名.方法()
```

- ArrayList 集合
  - 它属于引用数据类型（类）。我们可以看作一个长度可变的数组。
  - 创建集合的方式

```
ArrayList<要存储元素的数据类型> 变量名 = new ArrayList<要存储元素的数据类型>();
```

  - 集合中的常用方法
    - boolean add（Object obj）
    - Object get（int index）
    - int size（）
    - boolean add（int index， Object obj）
    - Object set（int index, Object obj）
    - Object remve（int index）
    - void clear（）