1、课程名称：范例：继承的应用


MLDN JAVA培训先锋

JAVA 应用开发详解

第6章：面向对象（高级）
—— 范例：继承的应用

MLDN 软件教学研发部

2、实例要求


实例要求

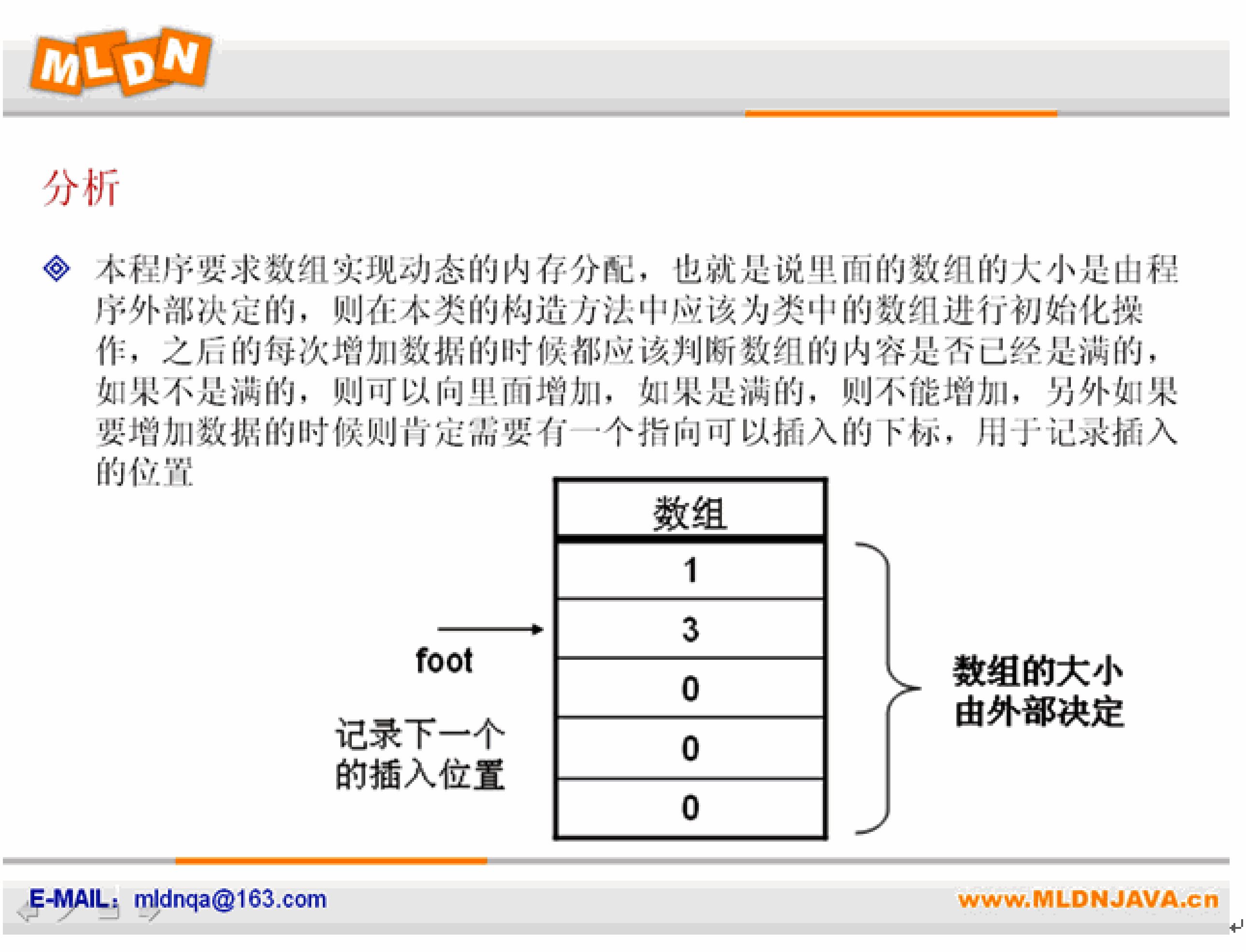
- 定义一个整型数组类，要求包含构造方法，增加数据及输出数据成员方法，并利用数组实现动态内存分配。在此基础上定义出以下子类：
  - A、 排序类 —— 实现排序
  - B、 反转类 —— 实现数据反向存放

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn

3、本实例主要采用的知识

继承的概念

4、具体内容


分析

- 本程序要求数组实现动态的内存分配，也就是说里面的数组的大小是由程序外部决定的，则在本类的构造方法中应该为类中的数组进行初始化操作，之后的每次增加数据的时候都应该判断数组的内容是否已经是满的，如果不是满的，则可以向里面增加，如果是满的，则不能增加，另外如果要增加数据的时候则肯定需要有一个指向可以插入的下标，用于记录插入的位置

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn

```
class Array{                // 表示数组
        private int temp[] ;            // 整型数组
        private int foot ; // 定义添加位置
        public Array(int len){
                if(len>0){
                        this.temp = new int[len] ;
                }else{
                        this.temp = new int[1] ;        // 最少维持空间是 1 个
                }
        }
```

```
        public boolean add(int i){      // 增加元素
                if(this.foot<this.temp.length){         // 还有空间
                        this.temp[foot] = i ;   // 增加元素
                        this.foot ++ ;// 修改脚标
                        return true ;
                }else{
                        return false ;
                }
        }
        public int[] getArray(){
                return this.temp ;
        }
};
class SortArray extends Array{      // 排序类
        public SortArray(int len){
                super(len) ;
        }
        public int[] getArray(){        // 覆写方法
                java.util.Arrays.sort(super.getArray()) ; // 排序操作
                return super.getArray() ;
        }
};
class ReverseArray extends Array{         // 反转操作类
        public ReverseArray(int len){
                super(len) ;
        }
        public int[] getArray() {
                int t[] = new int[super.getArray().length] ;       // 开辟一个新的数组
                int count = t.length - 1 ;
                for(int x=0 ;x<t.length;x++){
                        t[count] = super.getArray()[x] ;      // 数组反转
                        count-- ;
                }
                return t ;
        }
};
public class ArrayDemo{
        public static void main(String args[]){
```

```
                ReverseArray a = null ;         // 声明反转类对象
                a = new ReverseArray(5) ;  // 开辟 5 个空间大小
                System.out.print(a.add(23) + "\t") ;
                System.out.print(a.add(21) + "\t") ;
                System.out.print(a.add(2) + "\t") ;
                System.out.print(a.add(42) + "\t") ;
                System.out.print(a.add(5) + "\t") ;
                System.out.print(a.add(6) + "\t") ;
                print(a.getArray()) ;

        }
        public static void print(int i[]){      // 输出数组内容
                for(int x=0;x<i.length;x++){
                        System.out.print(i[x] + "、") ;
                }
        }
};
```

排序类，直接修改使用的子类即可：

```
public class ArrayDemo{
        public static void main(String args[]){
                // ReverseArray a = null ;      // 声明反转类对象
                // a = new ReverseArray(5) ; // 开辟 5 个空间大小
                SortArray a = null ;
                a = new SortArray(5) ;
                System.out.print(a.add(23) + "\t") ;
                System.out.print(a.add(21) + "\t") ;
                System.out.print(a.add(2) + "\t") ;
                System.out.print(a.add(42) + "\t") ;
                System.out.print(a.add(5) + "\t") ;
                System.out.print(a.add(6) + "\t") ;
                print(a.getArray()) ;

        }
        public static void print(int i[]){      // 输出数组内容
                for(int x=0;x<i.length;x++){
                        System.out.print(i[x] + "、") ;
                }
        }
```

```
};
```

5、总结

应用了继承的各个概念，包括覆写，子类对象的实例化过程，排序操作，可以发现使用继承可以让代码可以得到重用。


下一章内容

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn