

# 《Java 程序设计》实验报告二

学生姓名：李季鸿

班级：2021 级计本（3）班

学号：20213002624

实验地点：理工楼 601 室

指导教师：张春元

实验日期：2023-02-27

共 2 学时

实验环境：Win10+JDK1.8+集成化开发工具：IntelliJ IDEA

## 1. 实验目的

学习 Java 基本数据类型、数组和分支语句的应用，掌握集成化开发工具的使用。

## 2. 实验内容

- （1）用集成化开发工具完成实验教材 P9 实验 1 内容。
- （2）用集成化开发工具完成实验教材 P10 实验 2 内容。
- （3）用集成化开发工具完成实验教材 P12 实验 3 内容。
- （4）用集成化开发工具完成实验教材 P15 实验 1 内容。

## 3. 实验过程

报告撰写具体要求：截屏显示或直接写出实验 1 至实验 4 的源码和运行结果，并完成各实验后的练习（不用抄题，直接写答案即可）。

实验内容（1）：

```
1. public class GreekAlphabet
2. {
3.     public static void main (String args[ ])
4.     {
5.         int startPosition = 0, endPosition = 0;
6.         char cStart = 'α', cEnd = 'ω';
7.         startPosition = (int)cStart; //cStart 做 int 型转换运算，并将结果赋值给
            startPosition
8.         endPosition = (int)cEnd; //cEnd 做 int 型转换运算，并将结果赋值给 endPosition
9.         System.out.println("希腊字母\ 'α\ ' 在 unicode 表中的顺序位置:" + (int)cStart);
10.        System.out.println("希腊字母表: ");
11.        for(int i = startPosition; i <= endPosition; i++)
12.        {
13.            char c = '\0';
14.            c = (char)i; //i 做 char 型转换运算，并将结果赋值给 c
15.            System.out.print(" " + c);
16.            if((i - startPosition + 1) % 10 == 0)
17.                System.out.println("");
18.        }
19.    }
20. }
```

希腊字母'a'在unicode表中的顺序位置:945

希腊字母表:

α β γ δ ε ζ η θ ι κ

λ μ ν ξ ο π ρ ς σ τ

υ φ χ ψ ω

进程已结束,退出代码0

图 2.1 输出希腊字母表

实验 1 练习题:

(1) 答: 会提示: java: 不兼容的类型: 从 double 转换到 float 可能会有损失。

(2) 答: 不会, 会提示: java: 不兼容的类型: 从 double 转换到 float 可能会有损失, 需要注意, java 中 float 类型定义时, 必须要在浮点数后面加上 f 或者 F。

(3) 答: 不会, 会提示: java: 不兼容的类型: 从 int 转换到 byte 可能会有损失, 需要注意, java 中 byte 类型定义时, 必须要注意 Byte 类型数据的范围是:  $-2^7 \sim 2^7 - 1$ , 所以赋值 128 肯定会显示错误。

第二种情况, z 输出是-128, 在意料之中。

实验内容 (2):

```
1. public class InputArray {
2.     public static void main(String[] args) {
3.         int[] a = {100, 200, 300};
4.         //输出数组 a 的长度
5.         System.out.println("数组 a 的长度="+a.length);
6.         //输出数组 a 的引用
7.         System.out.println("数组 a 的位置="+a);
8.         int b[][] = {{1}, {1,1}, {1,2,1}, {1,3,3,1}, {1,4,6,4,1}}
9.         ;
10.        //输出二维数组 b 的一维数组的个数
11.        System.out.println(b.length);
12.        System.out.println(b[4][2]);
13.        //将数组 a 的引用赋给 b[4];
14.        b[4]=a;
15.        System.out.println(b[4][2]);
16.    }
```

```
数组a的长度=3
数组a的位置=[I@10f87f48
5
6
300
```

图 2.2 输出数组的引用和元素的值

实验 2 练习题:

(1) 答: 会提示: Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3 at InputArray.main(InputArray.java:21)超出了定义的数组长度。

编译没错, 运行时会出错。

(2) 答:

```
System.out.println("-----");
for(int i=0;i<b.length;i++){
    System.out.println(b[i].length);
    System.out.println(b[i]);
}
//将数组a的引用赋给b[4];
```

```
-----
1
[I@b4c966a
2
[I@2f4d3709
3
[I@4e50df2e
4
[I@1d81eb93
5
[I@7291c18f
```

(3) 答:

```
//将数组a的引用赋给b[4];
b[4]=a;
System.out.println("-----");
for(int i=0;i<b.length;i++){
    System.out.println(b[i].length);
    System.out.println(b[i]);
}
```

```
-----
1
[I@b4c966a
2
[I@2f4d3709
3
[I@4e50df2e
4
[I@1d81eb93
3
[I@10f87f48
```

实验内容 (3):

```
1. import java.util.Arrays;
2.
3. public class CopyArray {
4.     public static void main (String args[ ]) {
5.         int [] a = {1,2,3,4,500,600,700,800};
6.         int [] b,c,d;
7.         System.out.println(Arrays.toString(a));
8.         b = Arrays.copyOf(a,a.length);
```

```

9.         System.out.println(Arrays.toString(b));
10.        c = Arrays.copyOf(a, 4); //Arrays 调用 copyOf 方法复制数组 a 的
    前 4 个元素
11.        System.out.println(Arrays.toString(c)); //Arrays 调用
    toString 方法返回数组 c 元素值的字符串
12.        d = Arrays.copyOfRange(a, 4, a.length); //Arrays 调用
    copyOfRange 方法复制数组 a 的后 4 个元素
13.        System.out.println(Arrays.toString(d));
14.        c[c.length-1] = -100; //将-100 赋给数组 c 的最后一个元素
15.        d[d.length-1] = -200;
16.        System.out.println(Arrays.toString(a));
17.    }
18.}

```

D:\java\_codes\week2\out\production\week2 CopyArray

```

[1, 2, 3, 4, 500, 600, 700, 800]
[1, 2, 3, 4, 500, 600, 700, 800]
[1, 2, 3, 4]
[500, 600, 700, 800]
[1, 2, 3, 4, 500, 600, 700, 800]

```

图 2.3 输出、复制数组的元素

实验 3 练习题:

(1) 答:

```

c[c.length-1] = -100; //将-100赋给数组c的最后一个元素
System.out.println("-----");
int [] tom = Arrays.copyOf(c, newLength: 6);
System.out.println(Arrays.toString(tom));
System.out.println("-----");

```

```

-----
[1, 2, 3, -100, 0, 0]
-----

```

(2) 答: 参数输入有问题, 查看说明手册发现, 第三个参数应该是新的数据类型, 输入 8 肯定会报错的。

```
System.out.println("-----");
int [] jerry = Arrays.copyOf(d, newLength: 1, newType: 8);
System.out.println(Arrays.toString(jerry));
System.out.println("-----");
```

D:\java\_codes\week2\src\CopyArray.java:22:30

java: 对于copyOf(int[],int,int), 找不到合适的方法

方法 java.util.Arrays.<T>copyOf(T[],int)不适用

(无法推断类型变量 T


(实际参数列表和形式参数列表长度不同))


方法 java.util.Arrays.<T,U>copyOf(U[],int,java.lang.Class<? extends T[]>)


不适用

(无法推断类型变量 T,U

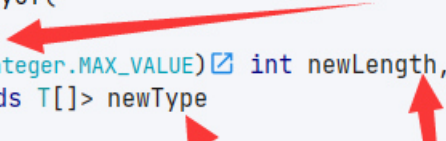
(参数不匹配; int无法转换为java.lang.Class<? extends T[]>))

 java.util.Arrays

 @NotNull

 @Contract(value = "\_,\_,null->new", pure = true)

```
public static <T, U> T[] copyOf(
    @NotNull U[] original,
    @Range(from = 0, to = Integer.MAX_VALUE) int newLength,
    @NotNull Class<? extends T[]> newType
)
```



Copies the specified array, truncating or padding with nulls (if necessary) so the copy has the specified length. For all indices that are valid in both the original array and the copy, the two arrays will contain identical values. For any indices that are valid in the copy but not the original, the copy will contain null. Such indices will exist if and only if the specified length is greater than that of the original array. The resulting array is of the class newType.

形参: original – the array to be copied  
 newLength – the length of the copy to be returned  
 newType – the class of the copy to be returned

返回值: a copy of the original array, truncated or padded with nulls to obtain the specified length

抛出: [NegativeArraySizeException](#) – if newLength is negative  
[NullPointerException](#) – if original is null  
[ArrayStoreException](#) – if an element copied from original is not of a runtime type that can be stored in an array of class newType

#### 实验内容 (4):

```
1. import java.util.Scanner;
2. public class Number {
3.     public static void main(String args[])
4.     {
5.         Scanner scanner=new Scanner(System.in);
6.         int number,d5,d4,d3,d2,d1;
7.         System.out.println("输入一个 1 至 99999 之间的数:");
8.         number=scanner.nextInt();
9.         if(1<number&&number<99999) //判断 number 在 1 至 99999 之间的
           条件
10.        {
11.            d5=number/10000;//计算 number 的最高位 (万位) d5
12.            d4=number/1000%10;    //计算 number 的千位 d4
13.            d3=number/100%10;    //计算 number 的百位 d3
14.            d2=number%100/10;
15.            d1=number%10;
16.            if(d5>0) //判断 number 是 5 位数的条件
17.            {
18.                System.out.println(number+"是 5 位数");
19.                if(d5==d1&&d2==d4) //判断 number 是回文数的条件
20.                {
21.                    System.out.println(number+"是回文数");
22.                }
23.                else
24.                {
25.                    System.out.println(number+"不是回文数");
26.                }
27.            }
28.            else if(d4>0) //判断 number 是 4 位数的条件
29.            {
30.                System.out.println(number+"是 4 位数");
31.                if(d1==d4&&d2==d3) //判断 number 是回文数的条件码
32.                {
33.                    System.out.println(number+"是回文数");
34.                }
35.                else
36.                {
37.                    System.out.println(number+"不是回文数");
38.                }
39.            }
40.            else if(d3>0) //判断 number 是 3 位数的条件
```

```

41.      {
42.          System.out.println(number+"是 3 位数");
43.          if(d1==d3) //判断 number 是回文数的条件
44.          {
45.              System.out.println(number+"是回文数");
46.          }
47.          else
48.          {
49.              System.out.println(number+"不是回文数");
50.          }
51.      }
52.      else if(d2!=0)
53.      {
54.          System.out.println(number+"是 2 位数");
55.          if(d1==d2)
56.          {
57.              System.out.println(number+"是回文数");
58.          }
59.          else
60.          {
61.              System.out.println(number+"不是回文数");
62.          }
63.      }
64.      else if(d1!=0)
65.      {
66.          System.out.println(number+"是 1 位数");
67.          System.out.println(number+"是回文数");
68.      }
69.  }
70.  else
71.  {
72.      System.out.printf("\n%d 不在 1 至 99999 之间",number);
73.  }
74.  }
75. }
76.

```

输入一个1至99999之间的数： 输入一个1至99999之间的数：

9889

9889是4位数

9889是回文数

12356

12356是5位数

12356不是回文数

### 实验 1 练习题:

(1) 答:

```
输入一个1至99999之间的数:  
2332  
2332是4位数  
2332是回文数
```

(2) 答:

```
输入一个1至99999之间的数:  
654321  
  
654321不在1至99999之间
```

(3) 答:

```
输入一个1至99999之间的数:  
33321  
33321是5位数  
33321不是回文数
```

### 4. 实验总结

写出实验中的心得体会 (对第 2 章理论课重点简述)。

答:

1. 标识符由字母、下划线、美元符号和数字组成, 并且第一个字符不能是数字字符。
2. 在 Java 语言中有 8 种基本数据类型, 即 `boolean`、`byte`、`short`、`int`、`long`、`float`、`double`、`char`。它们都有固定的大小和取值范围。注意在声明基本数据类型时需要指定其数据类型, 并且千万不能越界。
3. 数组是相同类型的数据元素按顺序组成的一种复合数据类型, 数组属于引用型变量, 因此两个相同类型的数组如果具有相同的引用, 它们就有完全相同的元素。声明数组时需要指定数组的类型和大小, 例如: `int[] array = new int[10]`; 其中 `array` 存放的是数组的首地址。
4. Java 中的基本数据类型和数组都是值传递。这意味着将基本数据类型或数组传递给方法时, 方法接收到的是它们的值的副本, 而不是它们的引用。因此, 方法中对基本数据类型或数组的修改不会影响原始值或数组。
5. 在 Java 中, 数组的索引从 0 开始。访问数组元素时要确保索引不会越界, 否则将抛出 `ArrayIndexOutOfBoundsException` 异常。
6. 基本数据类型和数组都可以使用运算符进行操作。例如, 可以对两个 `int` 类型的值进行加法操作, 也可以对两个 `int` 类型的数组进行加法操作。
7. 注意基本数据类型和数组在内存中的存储方式。基本数据类型在栈中存储, 而数组在堆中存储。这意味着基本数据类型的访问速度比数组更快, 但数组的大小可以动态分配。
8. Java 中还有一些包装类, 例如 `Integer`、`Double` 等, 它们用于将基本数据类型转换为对



象类型。这些包装类提供了一些便捷的方法来处理基本数据类型。例如，`Integer` 类提供了将字符串转换为 `int` 类型的方法 `parseInt()`。

9. 当声明一个数组时，需要注意数组的大小不能为负数，否则会抛出 `NegativeArraySizeException` 异常。

10. 在 Java 中，数组是可变长度的，可以通过重新分配数组大小来增加或减少数组的长度。要注意的是，重新分配数组大小会导致原来的数组元素被复制到新的数组中，因此可能会影响性能。

11. 数组的长度可以通过 `length` 属性获取，例如：`int[] arr = new int[10];`  
`int length = arr.length;` 注意，这里的 `length` 是没有括号的。