计算机学院

北華航天工業學院
NORTH CHINA INSTITUTE OF AEROSPACE ENGINEERING

第4章 数组与字符串

2021-2022-1

本章目标

- 数组
- Arrays类的使用
- 字符串操作

数组

◆数组的初始化

利用new 来为数组型变量分配内存空间

◆语法

数组名=new　数组元素类型[元素个数]

所有数组会自动初始化，基本数据类型值为：0,0.0f,false,对象为：null

一维数组

- 一维数组的使用

```
int array[];
array = new int[5];
array[index]; //例如array[0]=20
```

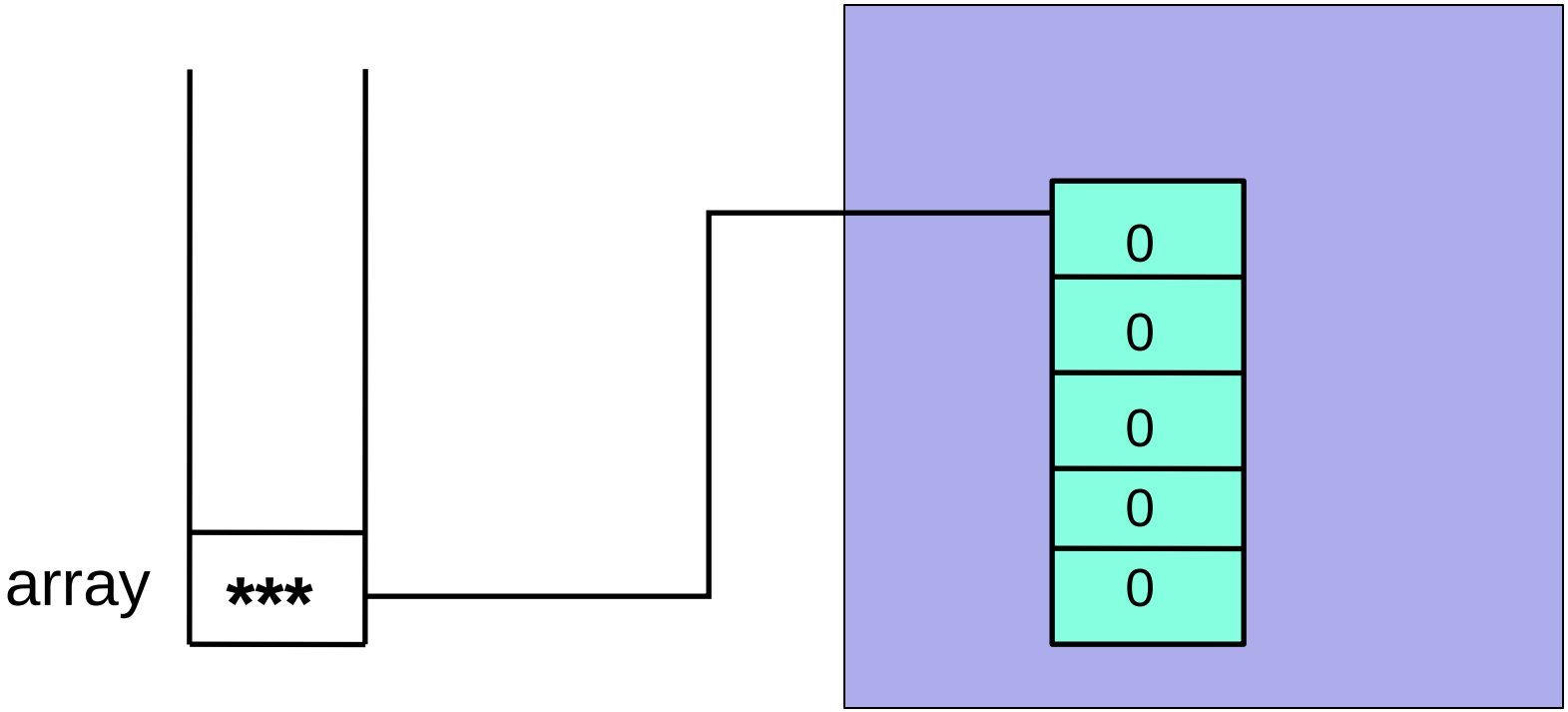
- index为数组下标，它可以为整型常数或表达式
- 数组的下标从0开始
- 每个数组都有一个属性length指明它的长度

  例如：array.length


```
public class Test{
  public static void main(String[] args) {
    int [] array
     array = new int[5];
    for (int i = 0; i < array.length; i++) {
          array[i] = i;
    }
  }
}
```

数组

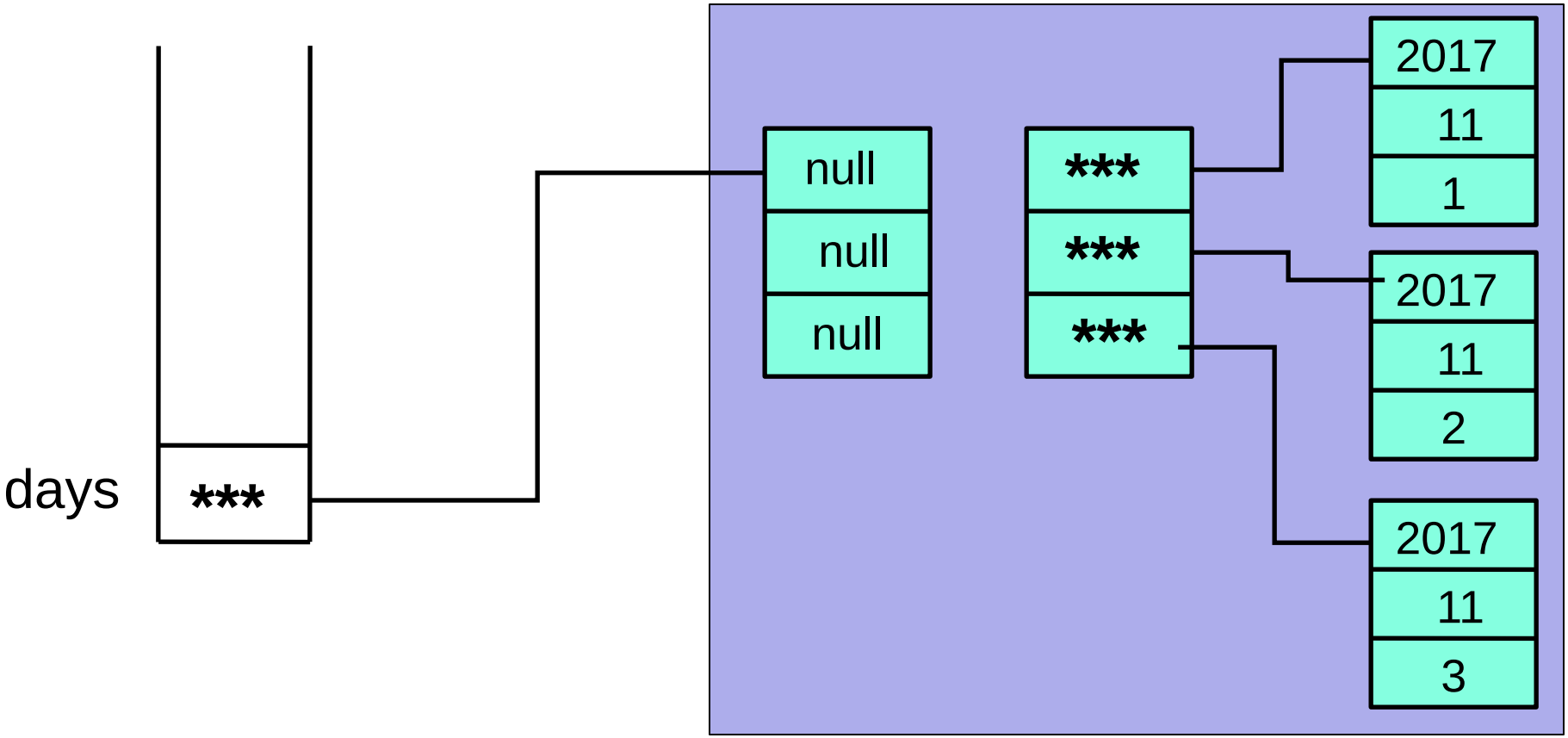
◆数组内存分配



```
public class Test{
  public static void main(String[] args) {
    Date [] days
     days = new Date[3];
    for (int i = 0; i < 3; i++) {
          days[i] = new Date(2017,11,i+1);
    }
  }
}
Class Date{
     int year; int month; int day;
      Date(int y, int m, int d) {
        year = y; month = m; day = d;
}
}
```

数组

◆数组内存分配



字符串操作类

- 两个类
  - java.lang.String类
  - java.lang.StringBuffer类
- 不同的应用场合

字符串操作类

- java.lang.String类—字符串/不可变字符序列
  - 构造方法
    - public **String**()
    - public **String**(byte bytes[])
    - public **String**(byte bytes[], int offset, int count)
    - public **String**(char value[])
    - public **String**(char value [], int offset, int count)
    - public **String**(String original)

字符串操作类

- java.lang.String类—字符串/字符序列
  - ◆构造方法的使用

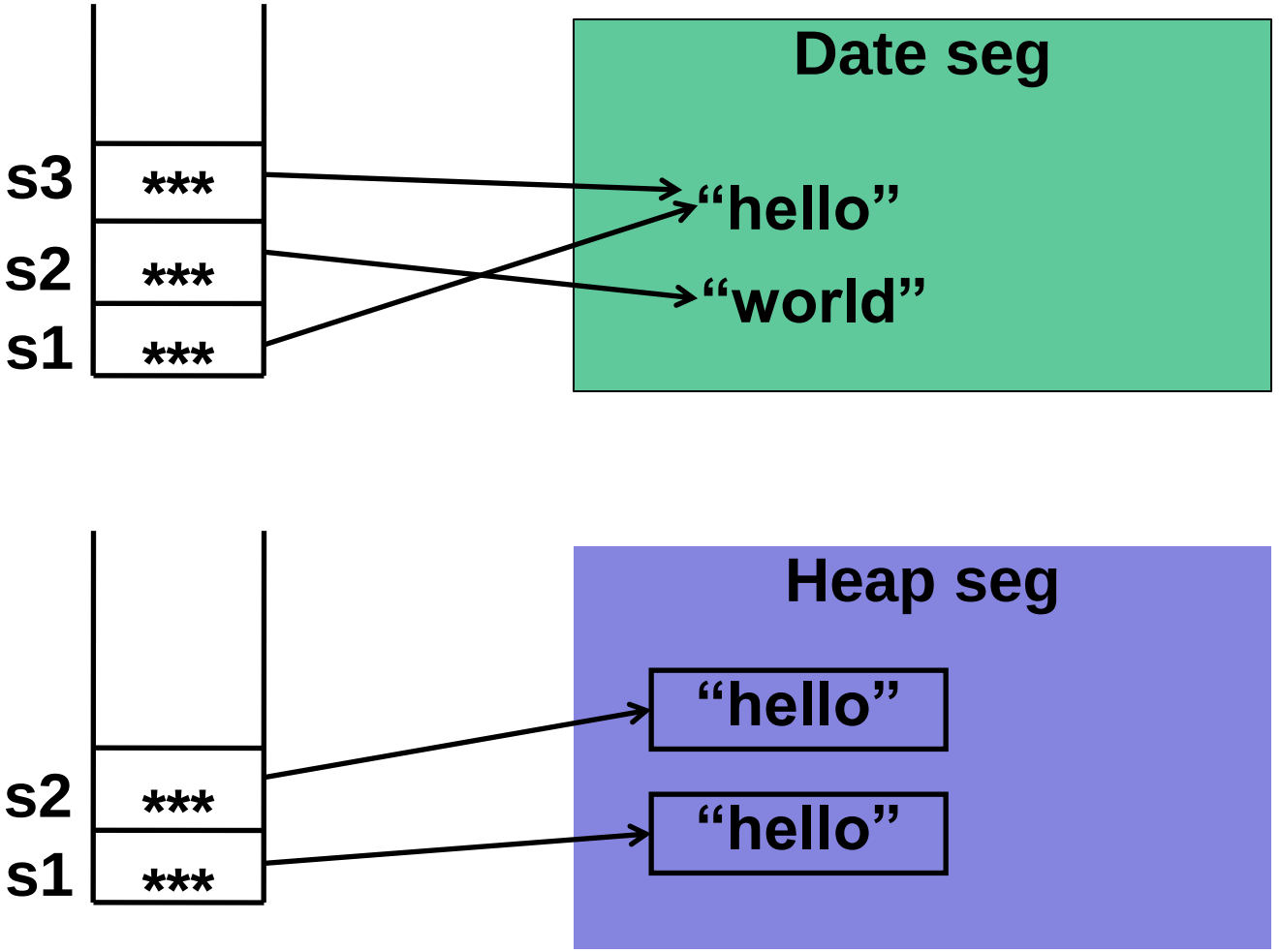
```
String s = new String();

char c[] = {'a', 'b', 'c'};
String s = new String(c);

char c[] = {'语', '言'};
String s = new String(c);
```

```
byte b[] = {97, 98, 99};
String s = new String(b);

String s = "abc";

String s = "语言";
```

字符串例子

```
public class Test {
    public static void main (String [] args) {
        String s1 = “hello”; String s2 = “world”;
        String s3 = “hello”;
        System.out.println( s1==s3);  //true
        s1 = new String (“hello”);
        s2 = new String (“hello”);
        System.out.println(s1 == s3); //false
        System.out.println(s1.equals(s2)); //true
        char c[] = { ‘s’,’ u’,’ n’,’ j’,’ a’,’ v’,’ a’ };
        String s4 = new String(c);
        String s5 = new String(c,4,4);
        System.out.println(s4);  //sun java
        System.out.println(s5);   //java
    }
}
```

字符串例子



String 类常用方法

public char charAt(int index)

返回字符串中第index个字符

public int length()

返回字符串长度

public int indexOf(String str)

返回字符串中出现str的第一个位置

public int indexOf(String str, int fromIndex)

返回字符串中从fromIndex开始出现str的第一个位置

public boolean equalsIgnoreCase(String another)

比较字符吕与another是否一样（忽略大小写）

public String replace(char oldChar, char newChar)

在字符串中用newChar字符替换oldChar字符

```
public class Test {
    public static void main ( String[] args ) {
      String s1 = “sun java”, s2 = “Sun Java”;
      System.out.println(s1.charAt(1));  //u
      System.out.println(s2.length());   //8
      System.out.println(s1.indexOf(“java”)); //4
      System.out.println(s1.indexOf(“Java”));  //-1
      System.out.println(s1.equals(s2));    //false
      System.out.println(s1.equalsIgnoreCase(s2));   //true

     String s = “我是程序员，我在学java”;
     String sr = s.replace(‘我’,’你’);
     System.out.println(sr); // 你是程序员，你在学java
    }
}
```

public boolean startsWith(String prefix)

判断字符串是否以prefix字符串开头

public boolean endsWith(String suffix)

判断字符串是否以suffix字符串结尾

public String toUpperCase()

返回一个字符串为该字符串的大写形式

public String toLowerCase()

返回一个字符串为该字符串的小写形式

public String substring(int beginIndex)

返回该字符串从beginIndex开始到结尾的子字符串

public String substring(int beginIndex, int endIndex)

返回该字符串从beginIndex开始到endIndex结尾的子字符串

public String trim()返回将该字符串去掉开头和结尾空格后的字符串

```
public class Test {
    public static void main ( String[] args ) {
      String s = “Welcome to Java World!” ;
      String s1 = “  sun  java  ” ;
      System.out.println(s.startWith( “Welcome” ));  //true
      System.out.println(s.endsWith( “World” ));   //false
      String sL = s.toLowerCase();
      String sU = s.toUpperCase();
      System.out.println(sL); //welcome to java world!
      System.out.println(sU); //WELCOME TO JAVA WORLD!
      String subs = s.substring(11);
      System.out.println(subs);    //Java World!
      String sp = s1.trim();
      System.out.println(sp);   //sun java
    }
}
```

String 类常用方法

- public static String valueOf(……)可以将基本类型数据转换为字符串；例如：

```
public   static   String valueOf( double d)
public   static   String valueOf( int i)
public   static    String valueOf(Object obj)
```

TestDateSort.java

- 方法 public String[] split(String regex)可以将一个字符串按照指定的分隔，返回分隔后的字符串数组

String 类常用方法

```
public  class   Test{
      public  static  void  main (String [] args) {
         int  j = 1234567;
         String  sNumber = String.valueof (j) ;
        System.out.println(“j 是”+sNumber.length()+“位数“);
         String  s = “Mary, F, 1976”;
         String[] split = s.split(“,”);
         for(int i =0;i< split.length;i++)  {
             System.out.println(split[i]);
         }
      }
 }
```

字符串操作类

- java.lang.StringBuffer类

  一个可变(mutable)/动态的字符序列

  构造方法

  ```
  public StringBuffer()
  public StringBuffer(int length)
  public StringBuffer(String str)
  ```

  主要方法

  添加(append)和插入(insert, 指定位置)

  ```
  public StringBuffer append(参数)
  public StringBuffer insert(int beginIndex, appendArguments)
  ```

  boolean, char, char[], double, float, int, long, String

  转换为字符串 - public String toString()

字符串操作类

- java.lang.StringBuffer类—方法举例

```
String s = "java语言";
StringBuffer buffer = new StringBuffer(s);
buffer.append(“easy”);
buffer.insert(6, “ is “);
s = buffer.toString();
System.out.println(s);
```

运行结果：
java语言 is easy.

字符串操作类

- java.lang.StringBuffer 代表**可变的字符序列**
- StringBuffer和String类似，但StringBuffer可以对其字符串进行改变。
- StringBuffer类的常见构造方法：
  - StringBuffer()

  创建一个不包含字符序列的“空”的StringBuffer对象。
  - StringBuffer(String str)

  创建一个StringBuffer对象，包含与String对象str相同的字符序列。

字符串操作类

- 重载方法 public StringBuffer append(……)可以为该StringBuffer对象添加字符序列，返回添加后的该StringBuffer对象引用，例如：

```
public StringBuffer append(String str)
public StringBuffer append(StringBuffer sbuf)
public StringBuffer append(char[] str)
public StringBuffer append(char[] str, int offset, int len)
public StringBuffer append( Object obj)
```

- 重载方法 public StringBuffer insert(……)可以为该StringBuffer对象在指定位置插入字符序列，返回修改后的该StringBuffer对象引用，例如：

```
public  StringBuffer insert(int offset,String str)
public  StirngBuffer insert (int offset,double d)
```

- 方法public StringBuffer delete(int start, int end)可以删除从start开始到end-1为止的一段字符序列，返回修改后的该StringBuffer对象引用。

字符串操作类

- 和String 类含义类似的方法：

```
public int indexOf(String str)
public int indexOf(String str, int fromIndex)
public String substring(int start)
public String substring(int start,int end)
public int length()
```

- 方法 public StringBuffer reverse()用于将字符序列逆序，返回修改后的该StringBuffer 对象引用。

字符串操作类

```
public class Test{
    public static void main( String[] args){
      String s = “Mircrosoft” ;
      char[] a = { ‘a’ ,’ b’ ,’ c’ };
      StringBuffer sb1 = new StringBuffer(s);
      sb1.append( ‘/’ ).append( “IBM” ).append( ‘/’ ).append( “Sun” );
      System.out.println(sb1);
      StringBuffer sb2 = new String;Buffer( “数字” ) ;
      for(int i = 0; i<=9; i++){sb2.append(i);}
       System.out.println(sb2);
      sb2.delete(8,sh2.length()).insert(0,a);
       System.out.println(sb2);
       System.out.println(sb2.reverse());
  }
}
```

字符串操作类

基本数据类型包装类

包装类（Integer,Double）这些类封装了一个相应的基本数据类型数值，并为其提供了一系列操作。

以java.lang.Integer类为例，构造方法：

```
Integer(int value)
Integer(String s)
```

字符串操作类

包装类常见的方法，以java.lang.Integer为例

public static final int MAX_VALUE最大的int型数

public static final int MIN_VALUE|最小的int型数

public long longValue() 返回封装数据的long型值

public double doubleValue() 返回封装数据的double型值

public int intValue() 返回封装数据的int型值

public static int pareseInt(String s) throws NumberFormatException

将字符串解析成int型数据，返回该 数值。

public static Integer valueof(String s)throws NumberFormatException

返回Integer对象，其中封装的整型数据为字符串S所表示。

```
public class Test{
    public static void main( String[] args){
        Integer i  = new  Integer(100);
        Double d  = new  Double(“123.456”);
        int j = i.intValue() + d.intValue();
        float f = i.floatValue() + d.floatValue();
        System.out.println(j);
        System.out.println(f);
        double pi = Double.parseDouble(“3.1415926”);
        double  r = Double.valueOf(“2.0”).doubleValue();
        System.out.println(s);
        try {
           int k = Integer.parseInt(“1.25”);
         }catch (NumberFormatException e){
             System.out.println(“数据格式不对“) ;
           }
}
```

命令行参数

- JAVA应用程序的主方法(程序的入口)

  public static void main (String args[]) {…}

  public static void main (String[] args) {…}

命令行参数

在启动JAVA应用程序时一次性地传递多个参数

C:\java 类名 参数 参数 ……

空格将参数分开

若参数包含空格，用双引号引起来

命令行参数

- 示例 1

```
C:\>java Test s1 s2
2
s1
s2

C:\>
```

```
C:\>java Test “s1 s2”
1
s1 s2

C:\>
```

```
C:\>java Test
0

C:\>
```

```
args) {
        System.out.println(len);
        for (int i = 0; i < len; i++)
            System.out.println(args[i]);
    }

}
```

命令行参数

- 示例 2

```
C:\>java Test s1 s2 s3
args[0]=s1
args[1]=s2
args[2]=s3

C:\>
```

```
class Test {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++)

        System.out.println(“args[“+i+”]=“+args[i]);
    }
}
```

总结

- 数组
  - 由类型相同的元素组成的有顺序的数据集合
  - 数组的长度固定，不能随意扩展
  - 可以存储任何数据类型
  - 数组中的一个元素可以通过它的下标来访问，下标从0开始
  - 数组分为一维数组和多维数组
- 字符串的常用操作

谢谢！