

Introduction to Java

آرایه ها و رشته ها

در مسائلی که تا حال بررسی کردیم، توسط یک متغیر تعدادی ورودی را دریافت می‌کردیم و اعمال لازم را روی ورودیها انجام می‌دادیم ولی همیشه با این مسائل روبرو نیستیم. فرض کنید بخواهیم اطلاعات 100 کارمند را از ورودی بخوانیم و سپس آنها را مرتب کنیم، در اینصورت باید ورودیها را در جایی از حافظه ذخیره کنیم. در زبانهای برنامه‌نویسی معمولاً از آرایه برای ذخیره اطلاعات در حافظه استفاده می‌کنند. در آرایه‌ها ما با توجه به تعداد ورودیها، طول آن را مشخص می‌کنیم. سپس داده‌ها را خوانده در آن قرار می‌دهیم.

Arrays in Java

- Java supports arrays
- An array is a collection of elements where each element is the same type.
 - Element type can be primitive or Object
 - Each element is a single value
 - The length of the array is set when it is created. It cannot change.
- Individual array elements are accessed via an index.
 - Array index numbering starts at 0.
- Note: Some references claim that arrays in Java are Objects. **THIS IS NOT TRUE.**
 - Arrays do exhibit some behaviour which is similar to objects, but they are not themselves, objects.

تعریف آرایه

خانه‌های پشت سر هم از حافظه، که هم‌نوع بوده و توسط یک اسم معرفی می‌شوند، آرایه نام دارد. نحوه دسترسی به هر یک از اعضاء آرایه، از طریق اندیس آرایه امکانپذیر است. برای تعریف آرایه ابتدا طول آرایه که در حقیقت تعداد خانه‌های آن را مشخص می‌کند، معین می‌کنیم. سپس نوع خانه‌ها باید معین شوند.

Creating Arrays

- Creating an array is a 2 step process
 - It must be declared (declaration does not specify size)

declaration syntax:

```
type[] arrayName;
```

 note the location of the []

- It must be created (ie. memory must be allocated for the array)

```
int[] grades;                // declaration

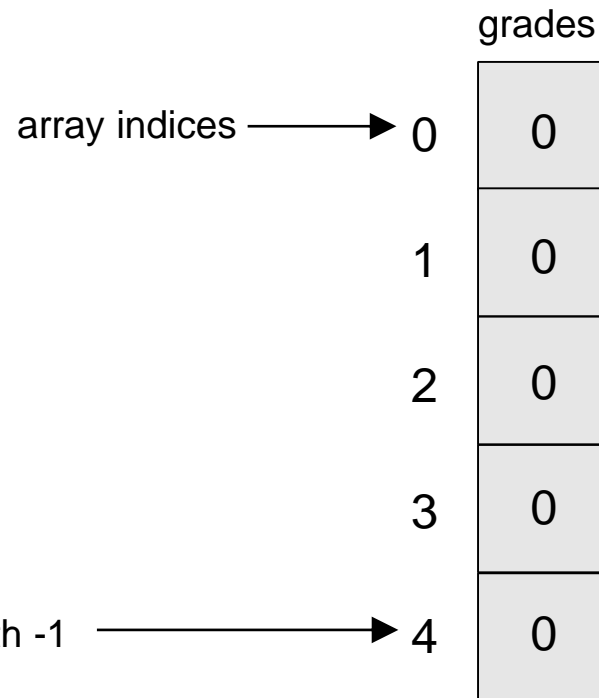
grades = new int[5];         // Create array.
                             // specify size
                             // assign new array to
                             // array variable
```

Creating Arrays

- When an array is created, all of its elements are automatically initialized
 - 0 for integral types
 - 0.0 for floating point types
 - false for boolean types
 - null for object types

```
int[] grades = new int[5];
```

Note: maximum array index is length -1



Initializing and Using Arrays

- Because array elements are initialized to 0, the array should be initialized with usable values before the array is used.
 - This can be done with a loop
 - Arrays have a length attribute which can be used for bounds checking
 - Elements are accessed using an index and []

```
int[] sequence = new int[5];  
  
for (int i=0; i< sequence.length; i++)  
{  
    sequence[i] = i * 25;  
}
```

Array element being accessed. In this case, it is being assigned a value.

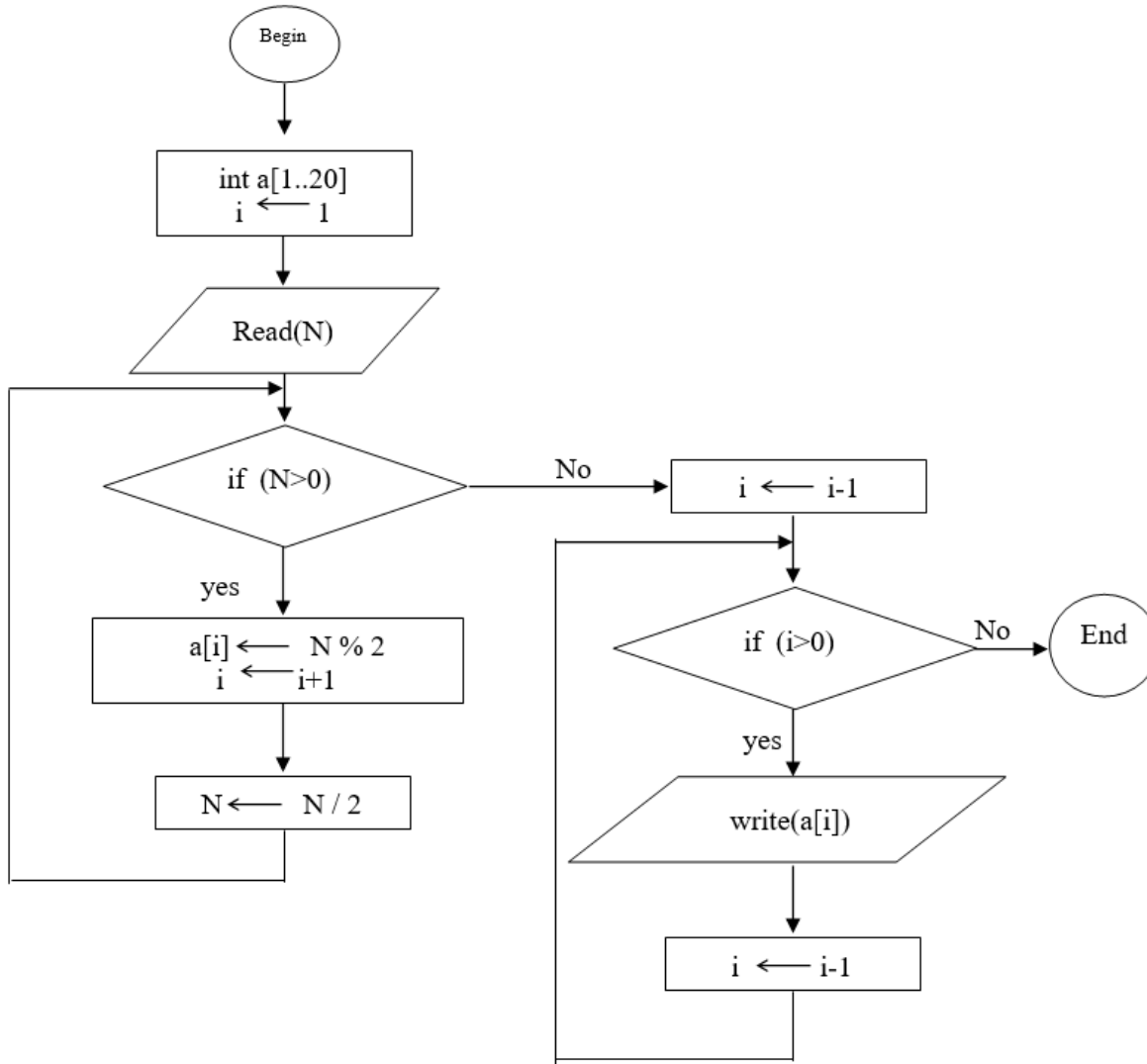
array length: ensures loop won't go past end of the array

Array Bounds Checking

- Whenever an array is accessed, the index is checked to ensure that it is within the bounds of the array.
- Attempts to access an array element outside the bounds of the array will cause an `ArrayIndexOutOfBoundsException` exception to be thrown.

```
int[] sequence = new int[5];  
  
sequence[0] = 50;           // ok  
sequence[1] = 60;           // ok  
sequence[-1] = 100;         // Exception  
sequence[5] = 30;           // Exception
```

برنامه ای بنویسید که عددی را از ورودی دریافت کرده آن را به مبنای ۲ برده.



فلوچارتی رسم نمائید که یک آرایه 100 عنصری را از ورودی دریافت کرده، سپس معکوس آن را به دست آورده، آرایه حاصل را در خروجی چاپ نماید.
فلوچارت بالا برای 6 عنصر بصورت زیر عمل می کند:

1	12	17	6	16	2
---	----	----	---	----	---



2	16	6	17	12	1
---	----	---	----	----	---

برنامه ای بنویسید که عددی از ورودی دریافت کرده سپس اعداد اول قبل از آن را تولید نموده ، در یک آرایه قرار دهد.

مثال

برنامه ای بنویسید که عددی از ورودی دریافت کرده سپس آن را به عامل های اول تجزیه نماید.

$$24=(2^3)*(3^1)$$

The String Class

- Although we haven't yet discussed classes and object, we will discuss the String class.
- String objects are handled specially by the compiler.
 - String is the only class which has "implicit" instantiation.
- The String class is defined in the java.lang package.
- Strings are immutable. The value of a String object can never be changed.
 - For mutable Strings, use the StringBuffer class.

Creating String Objects

- Normally, objects in Java are created with the *new* keyword.

```
String name;  
    name = new String("Craig");
```

- However, String objects can be created "implicitly":

```
String name;  
    name = "Craig";
```

- Strings can also be created using the + operator. The + operator, when applied to Strings means concatenation.

```
int age = 21;  
String message = "Craig wishes he was " + age + " years old";
```

Commonly used String methods

- The String class has many methods. The most commonly used are:
 - `length()` - returns the number of characters in the String
 - `charAt()` - returns the character at the specified index
 - `equals()` - returns true if two strings have equal contents
 - `compareTo()` - returns 0 if equal, -# if one String is "less than" the other, +# if one String is "greater than" the other.
 - `indexOf()` - returns the index of specified String or character
 - `substring()` - returns a portion of the String's text
 - `toUpperCase()`, `toLowerCase()` - converts the String to upper or lower case characters

String Examples

```
String name = "Craig";  
String name2 = "Craig";  
  
if (name.equals(name2))  
    System.out.println("The names are the same");
```

```
String name = "Craig Schock";  
int lastNameIndex = name.indexOf("Schock");
```

```
String grade = "B+";  
double gpa = 0.0;  
  
if (grade.charAt(0) == 'B')  
    gpa = 3.0;  
  
if (grade.charAt(1) == '+')  
    gpa = gpa + 0.3;
```

Testing Strings for Equality

- Important note: The `==` operator cannot be used to test String objects for equality
 - Variables of type String are references to objects (ie. memory addresses)
 - Comparing two String objects using `==` actually compares their memory addresses. Two separate String objects may contain the equivalent text, but reside at different memory locations.
- Use the `equals` method to test for equality.

مثال: برنامه ای بنویسید که رشته ای از ورودی دریافت کرده اعداد آنرا حذف نموده و رشته حاصل را چاپ کند.

```
String st2="";
System.out.println("original text==" + st);
for(int i=0; i<st.length();i++){
    if(!((st.charAt(i)>='0') && (st.charAt(i)<='9'))){
        st2+=st.charAt(i);
    }
}
System.out.println("Final text=" + st2);
```

مثال: برنامه ای بنویسید که رشته ای عددی را از ورودی دریافت کرده
آنرا به عدد تبدیل نموده عدد حاصل را چاپ کند.

آرایه ای از رشته ها

- `String [] nameOfString= new String[length]`

```
String[] rank = {  
    "2", "3", "4", "5", "6", "7", "8", "9",  
    "10", "Jack", "Queen", "King", "Ace"  
};
```

مثال: برنامه ای بنویسید که اطلاعات حداکثر ۱۰۰ دانشجو که عبارتند از:

اسم و فامیلی
شماره دانشجویی
معدل

از ورودی دریافت کرده سپس مشخصات دانشجویان با معدل الف را در خروجی چاپ نماید.

جستجو و مرتب سازی

یکی از مسائلی که در بحث طراحی الگوریتم بسیار مهم است، بحث مرتب سازی و جستجو می باشد. منظور از جستجو اینست که یک مقداری را از یک لیست جستجو کنیم و منظور از مرتب سازی اینست که یک لیست مرتب از داده ها را تولید کنیم.

برای جستجو و مرتب سازی الگوریتم های مختلفی وجود دارد در زیر الگوریتم های اولیه، برای جستجو و مرتب سازی را بررسی می کنیم.

همانطور که در بالا اشاره کردیم منظور از جستجو، یافتن عنصری در یک لیست می باشد.

دو الگوریتم زیر غالباً برای جستجو بکار می روند:

- جستجوی خطی linear search

- جستجوی دودویی binary search

Sorting Algorithms

- Bubble Sort
- Selection Sort
- Insertion Sort
- Quick Sort
- Merge Sort

Bubble Sort

```
for (int i=0; i<n; i++)  
    for (int j=0; j<n-1; j++)  
        if ( x [ j ] > x [ j + 1 ] )  
        {  
            temp = x [ j ] ;  
            x [ j ] = x [ j + 1 ] ;  
            x [ j + 1 ] = temp ;  
        }
```

Selection Sort

```
for (int i=0; i<n; i++)
{
    Min = x [ i ] ;
    index = i ;

    for( j = i +1 ; j<n;
        j++)
        if ( x [ j ] < Min)
        {
            Min = x [ j ] ;
            index = j ;
        } // find The smallest Element
    x [ index ] = x [ i ] ;
    x [ i ] = Min ; // swap Minimum With other Element
} // end of selection sort
```

مثال: برنامه ای بنویسید که اطلاعات حداکثر ۱۰۰ دانشجو که عبارتند از:

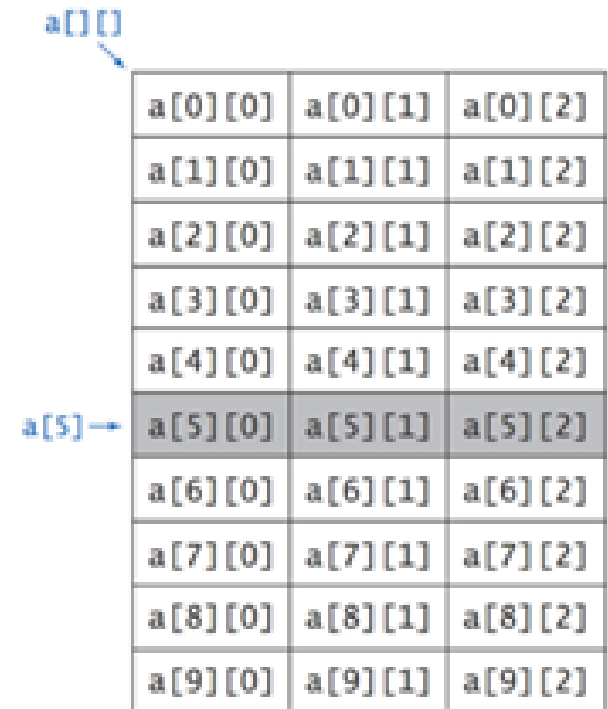
اسم و فامیلی
شماره دانشجویی
معدل

از ورودی دریافت کرده سپس مشخصات دانشجویان را بر اساس اسم و فایلی مرتب نموده در خروجی چاپ نماید.

آرایه های دو بعدی

type [][] name= new type[row][column]

```
int M = 10;  
int N = 3;  
double[][] a = new double[M][N];  
for (int i = 0; i < M; i++) {  
    for (int j = 0; j < N; j++) {  
        a[i][j] = 0.0;  
    }  
}
```



a[0][0]	a[0][1]	a[0][2]
a[1][0]	a[1][1]	a[1][2]
a[2][0]	a[2][1]	a[2][2]
a[3][0]	a[3][1]	a[3][2]
a[4][0]	a[4][1]	a[4][2]
a[5][0]	a[5][1]	a[5][2]
a[6][0]	a[6][1]	a[6][2]
a[7][0]	a[7][1]	a[7][2]
a[8][0]	a[8][1]	a[8][2]
a[9][0]	a[9][1]	a[9][2]

□ مقداردهی آرایه دو بعدی با لیست کردن مقادیر.

```
double[][] p = {  
    { .02, .92, .02, .02, .02 },  
    { .02, .02, .32, .32, .32 },  
    { .02, .02, .02, .92, .02 },  
    { .92, .02, .02, .02, .02 },  
    { .47, .02, .47, .02, .02 },  
};
```

	.92	.02	.02	.02	.02
سطر ۱ →	.02	.02	.32	.32	.32
	.02	.02	.02	.92	.02
	.92	.02	.02	.02	.02
	.47	.02	.47	.02	.02
			ستون ۳ ↑		

declaration syntax:

```
type[] [] arrayName;
```

each [] indicates another dimension

- ```
int grades = new int[20];
```

```
int[][] grades = new int[20][5];
for(int i = 0; i< 20; i++)
 for(int j = 0; j<5; j++)
 grades[i][j] = 100;
```

```
String[][] colours = {{ "Red", "Green", "Blue"},
 {"Cyan", "Magenta", "Yellow"},
 {"Russet", "Mauve", "Orange"}};
```

MyClass.java

```
public class MyClass {
 public static void main(String[] args) {
 int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
 int x = myNumbers[1][2];
 System.out.println(x);
 }
}
```

Result:

7

# جمع دو تا ماتریس

- برنامه ای بنویسید که دو ماتریس  $5 \times 5$  را از ورودی دریافت کرده مجموع دو ماتریس را جمع و در خروجی چاپ نماید.



# ضرب دو تا ماتریس

- برنامه ای بنویسید که دو ماتریس ضربپذیر را از ورودی دریافت کرده حاصل ضرب دو ماتریس را محاسبه و در خروجی چاپ نماید.

# The StringBuffer Class

---

- StringBuffer objects are similar to String objects
  - Strings are immutable
  - StringBuffers are mutable
- The StringBuffer class defines methods for modifying the String value
  - insert()
  - append()
  - setLength()
- To clear a StringBuffer, set it's length to 0

```
StringBuffer nameBuffer = new StringBuffer("Joe");
[...]
nameBuffer.setLength(0); // clear StringBuffer
```

# StringBuffer Example

---

```
StringBuffer sql = new StringBuffer();

sql.setLength(0);
sql.append("Select * from Employee");
sql.append(" where Employee_ID = " + employeeId);
sql.append(" and Employee_name = '" + employeeName + "'");
```