



# CSN-103: Fundamentals of Object Oriented Programming

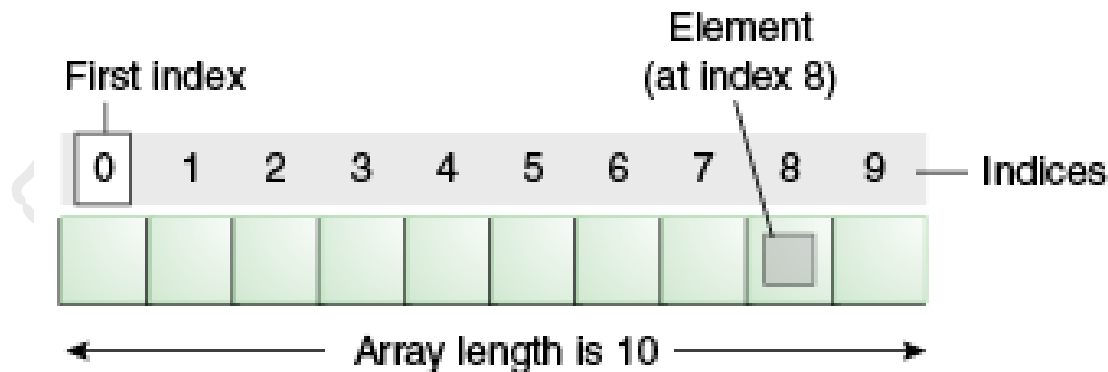
**Instructor: Dr. Rahul Thakur**

Assistant Professor, Computer Science and Engineering, IIT Roorkee



# Array

- An array is a collection of same type variables that are referred to by a common name
  - It is a **Data Structure**: It is a data organization, management, and storage format that enables efficient access and modification
- Arrays of **any type** can be created and may have **one or more** dimensions
- A specific element in an array is accessed by its **index**



- **Array Index vs. Array Size**

# Array

- Array declaration (Example: int Array)

```
int marks[]; // marks is an array variable, no memory allocated
```

- We must allocate memory using **new** keyword

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

OR 

```
int marks[] = new int[5];           //Automatic initialization to 0
```

- Accessing the element: Using array index (starts with 0)

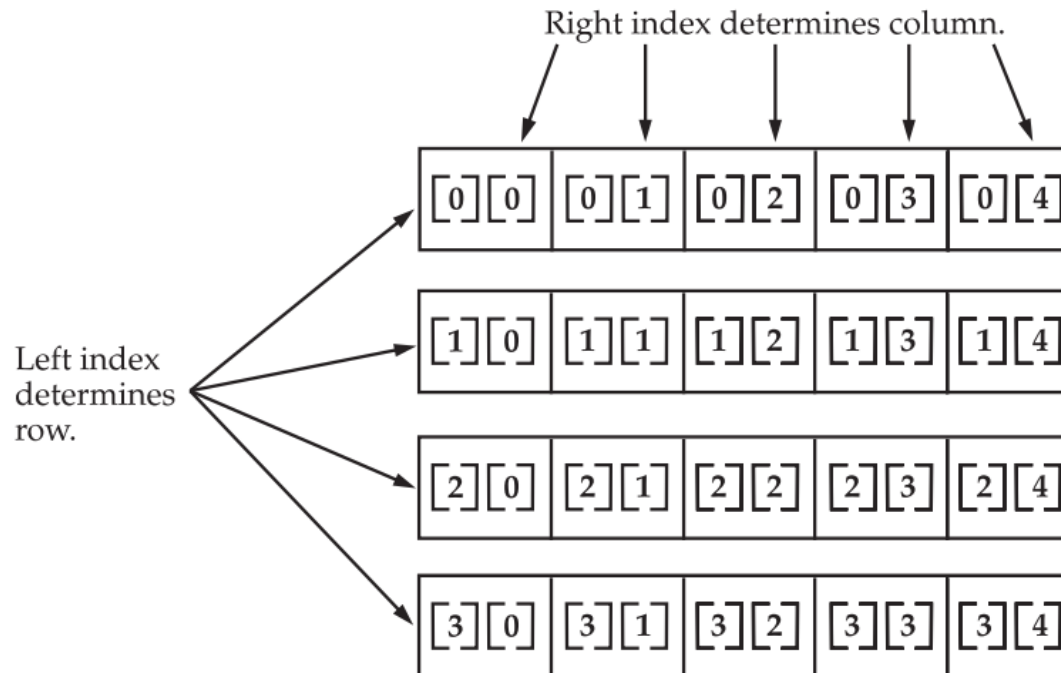
```
marks[1] = 60;           // Assigning a value
```

```
System.out.println(marks[3]); // Printing a value
```

# Multidimensional Arrays

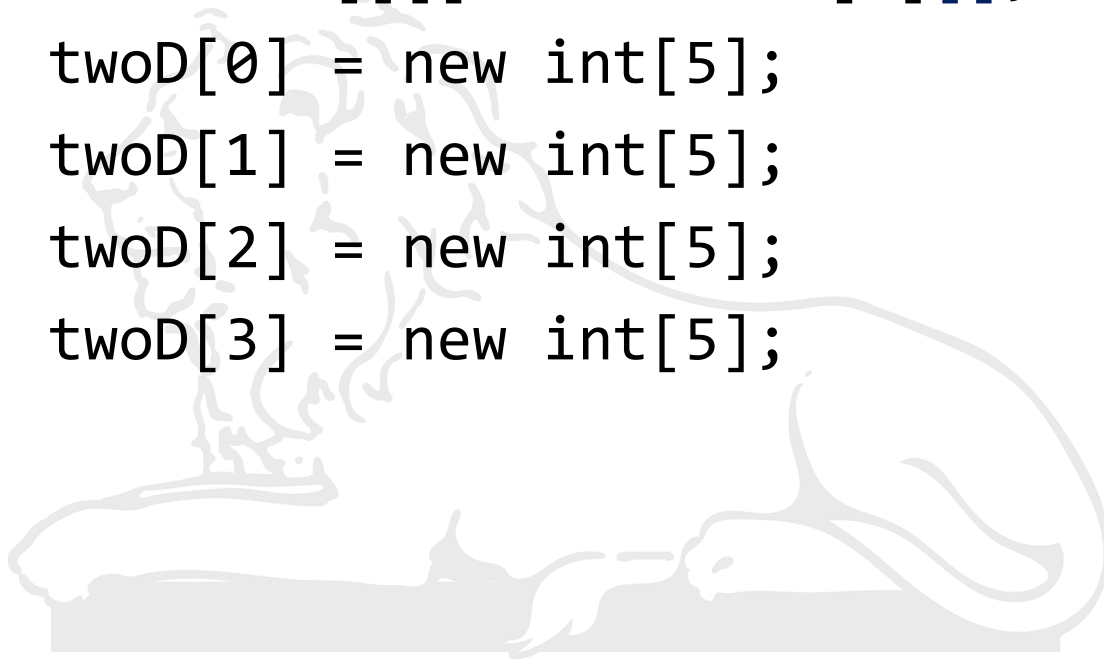
- Multidimensional arrays: Arrays of arrays
- **Example:** Declare a two-dimensional array variable

```
int twoD[][] = new int[4][5];
```



- For a multidimensional array, we only need to specify the size for the first (leftmost) dimension

```
int twoD[][] = new int[4][];  
twoD[0] = new int[5];  
twoD[1] = new int[5];  
twoD[2] = new int[5];  
twoD[3] = new int[5];
```



# Arrays

- No need to allocate the same number of elements for each dimension

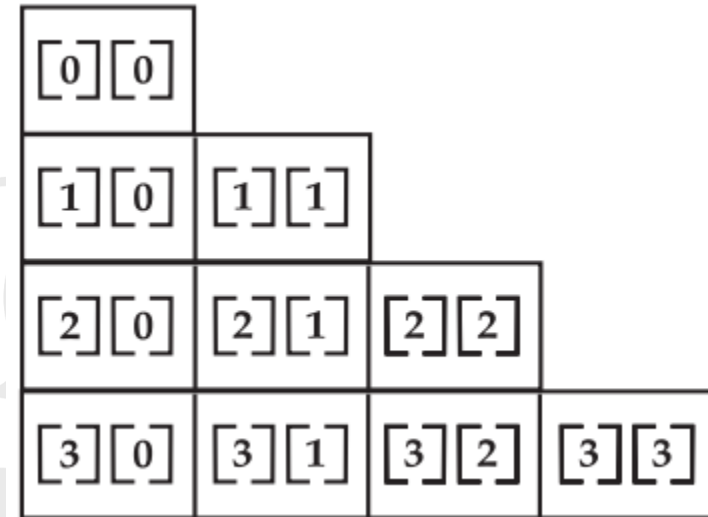
```
int twoD[][] = new int[4][];
```

```
twoD[0] = new int[1];
```

```
twoD[1] = new int[2];
```

```
twoD[2] = new int[3];
```

```
twoD[3] = new int[4];
```



# Alternative Array Declaration Syntax

- Examples:
  - ```
int a1[] = new int[3];  
int[] a1 = new int[3];  
int []a1 = new int[3];
```

Equivalent
  - ```
int a2[][] = new int[3][4];  
int[][] a2 = new int[3][4];  
int [][]a2 = new int[3][4];
```

Equivalent
  - ```
int[] a1, a2, a3;  
int a1[], a2[], a3[];  
int []a1, []a2, []a3;
```

Equivalent

# Array Length and Cloning

- **length**: A variable that contains the size of the array

- No need to declare, directly available for use

```
int a[] = new int[5];  
System.out.println(a.length);
```

```
int a1[][] = new int[3][4];  
System.out.println(a1.length);  
System.out.println(a1[0].length);
```

- Cloning and Printing: **clone()** and **toString()**

```
import java.util.Arrays;  
int ia1[] = { 1, 2, 3 };  
int ia2[] = ia1.clone();  
System.out.println(Arrays.toString(ia1));  
System.out.println(Arrays.toString(ia2));
```



# Strings

---

# Creating Strings

- A String can be created in many ways (**Constructors**)

- Empty string

```
String s = new String();
```

- String using a char array

```
char chars[ ] = { 'a', 'b', 'c' };
```

```
String s = new String(chars); // "abc"
```

- String using another String

```
String str = new String(s); // "abc"
```

- The **length of a string** is the number of characters that it contains

```
System.out.println(s.length());
```

# String Operations

- Creating String literal/constant

```
String s2 = "abc";
```

**//CAUTION: String Pool**

- String Concatenation with Other Data Types

```
int age = 9;
```

```
String s = "He is " + age + " years old";
```

```
System.out.println(s);
```

- Character Extraction

```
String s = "Hello World";
```

```
ch = s.charAt(0);
```

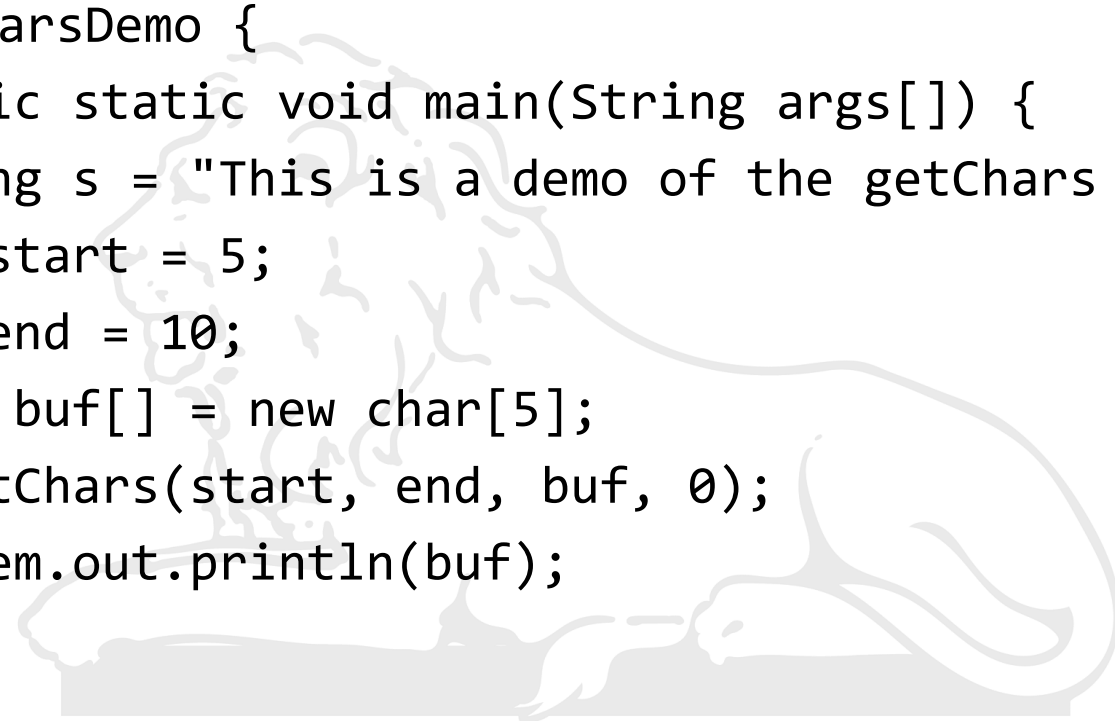
```
ch = "Hello World".charAt(0);
```

//First character at location/index 0

# String Operations

- To extract more than one character at a time: **getChars( )**

```
class getCharsDemo {  
    public static void main(String args[]) {  
        String s = "This is a demo of the getChars method";  
        int start = 5;  
        int end = 10;  
        char buf[] = new char[5];  
        s.getChars(start, end, buf, 0);  
        System.out.println(buf);  
    }  
}
```

A faint, stylized illustration of a lion lying down, serving as a background for the code block.

# String Operations

- To convert a String object into a character array
  - Use `toCharArray( )`
- String Comparison
  - For equality: `equals( )` and `equalsIgnoreCase( )`
- `equals( )` vs. `==`
  - `equals( )` method compares the characters inside a String object
  - `==` operator compares two object references to see whether they refer to the same instance

# Other String Operations: For Self Study

- `startsWith()` and `endsWith()`
- `compareTo()`
- `indexOf()` and `lastIndexOf()`
- `substring()`
- `concat()`
- `replace()` //Do not use for Assignment 1 and 2
- `trim()`
- `toLowerCase()` and `toUpperCase()`
- `isEmpty()`

