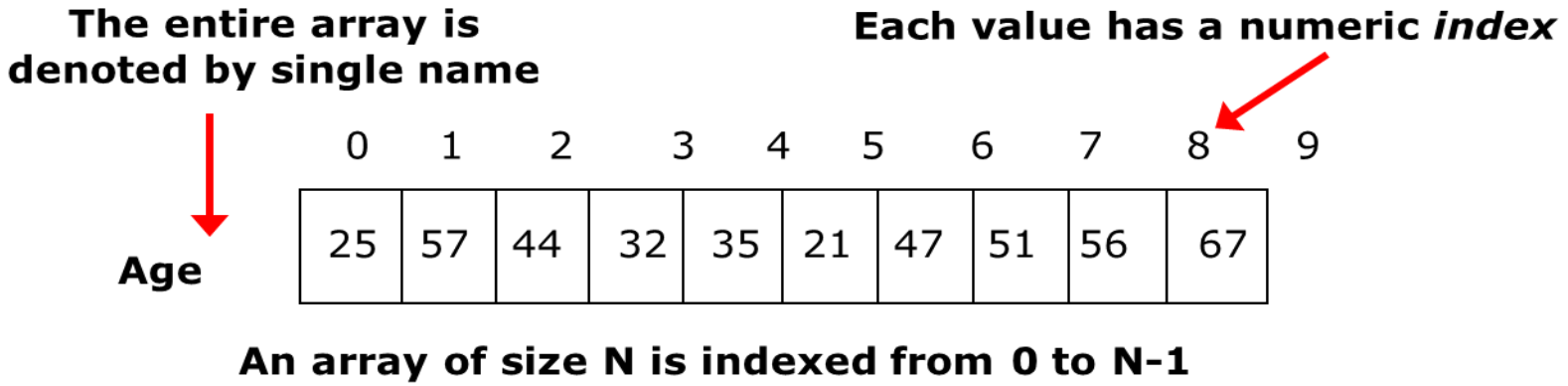
# Arrays

# Contents

# Introduction

- An **array** is a **container** that holds a **fixed number of values** of a **same data type.**
- **Need of Array: Difficult to manage** large number of variable in normal way. The **idea of array** is to **represent many instances in one variable.**
- Length of the **array is fixed** and index **starts with 0. Can't access** the elements **beyond the array limit.**



# Types

- There are **three types of arrays** in Java:
    - Single Dimensional Array
    - Multidimensional Array

# Single Dimensional Array

- **Single dimensional array** is a collection of items under **common name** in **linear array** fashion.
- **Declaration**
- **Syntax**

dataType[] arrayName; (or)

dataType []arrayName; (or)

dataType arrayName[];

**Instantiation of an Array in Java**

arrayReferenceVariable=**new** dataType[size]

**Example**

int []salary; ***//declaration***

salary=new int[10]; ***//Instantiation***

**Declaration and Instantiation**

int salary[]=**new** int[10];

**Note:**

- Array **declaration** only create **reference of array variable**.
- To **create or give memory** to **array only at** that **time of array instantiation**.

# Single Dimensional Array

## Create Array

- int [] marks=new int[10];  **//declaration and instantiation**

  **(OR)**

- int marks[]=new int[10];
- The **[] operator** is the **indication to the compiler** we are **naming an array object** and not an ordinary variable.
- Java Array is object, we use the **new operator** to create an array.

# Single Dimensional Array

**Define or initialize Array values:**

- There are **two ways** to define the array values:

1. **During Declaration**: int [] marks={89, 90, 67,35, 99, 80};
2. **Using index :** int marks[]=new int[10]; ***//* declaration**

   marks[0]=89;

   marks[1]=90;

   ……..

- **Array Length**: To find the array length java provides **inbuilt length variable**

  int len=marks.length; //**assigns the size of marks array in to the variable len**

# Single Dimensional Array

## Retrieving and Processing Array Elements

- Array elements are accessed using index value. The index starts with 0.

```
element=marks[3];  //retrieve 4th element of array

marks[4]= 100; // update 100 into 5th position
```

- Since more number of elements are there in an array, **loops are more convenient way** to process.
- In general, array elements are **accessed and processed** using **for loop** and **for….each loop**.

# Single Dimensional Array

```
/**
  * The ArrayDemo1 class implements an application that
  * Illustrate the access array elements
*/
 public class ArrayDemo1 {
     public static void main(String[] args) {
                int[] marks = new int[3]; ; // create array
                marks[0] = 89;  //assign values
                marks[1] = 90;
                System.out.println("Element at index 0: " + marks[0]); //access elements
                System.out.println("Element at index 1: " + marks[1]);
                System.out.println("Element at index 2: " + marks[2]);
     }
}
```

**Output**:

Element at index 0: 89

Element at index 1: 90

Element at index 2: 0

# Single Dimensional Array

```
/**
  * The CustomerDetail class implements an application that
  * Illustrate the access array elements
*/
 public class CustomerDetail {
public static void main(String[] args) {
            String[] CustName = new String[5]; ; // create array
            CustName[0] = "Aaron";  //assign values
            CustName[1] = "Kavin";
            CustName[2] = "Jesicca";
            CustName[3] = "Rishabh";
            CustName[4] = "Vinitha";
```

# Single Dimensional Array

```
            System.out.println("*********CUSTOMER DETAIL********");
            System.out.println(CustName[0]); //access elements
            System.out.println(CustName[1]);
            System.out.println(CustName[2]);
            System.out.println(CustName[3]);
            System.out.println(CustName[4]);
        }
}
```

**Output**:

```
*********CUSTOMER DETAIL********
Aaron
Kavin
Jesicca
Rishabh
Vinitha
```

# Single Dimensional Array

```
/**
  * The ArrayDemoForEach class implements an application that
  * Illustrate the access array elements using for-each statements
*/
Class ArrayDemoForEach {
     public static void main(String[] args)  {
                        int[] marks = {56, 84, 52, 90, 100};
                        System.out.println("Using for each Loop:");
                        for(int  i:marks) {
                              System.out.println(i);
                  }
        }
}
```

**Output**:

Using for each Loop:

56

84

52

90

100

# Single Dimensional Array

```
/**
  * The CustomerDetailForEach class implements an application that
  * Illustrate the access array elements using for-each statements
*/
public class CustomerDetailForEach {
      public static void main(String[] args) {
            String[] CustName = new String[5]; ; // create array
            CustName[0] = "Aaron";  //assign values
            CustName[1] = "Kavin";
            CustName[2] = "Jesicca";
            CustName[3] = "Rishabh";
            CustName[4] = "Vinitha";
```

# Single Dimensional Array

```
        System.out.println("**********CUSTOMER DETAIL**********");
        for(String Name : CustName) {
            System.out.println(Name);
        }
    }
}
```

**Output**:

```
**********CUSTOMER DETAIL**********
Aaron
Kavin
Jesicca
Rishabh
Vinitha
```

# Multi Dimensional Array

- In general, **more than one dimension** refer to the multi dimensional array. Its is **array of arrays**. The retrieve and processing like single dimensional array.
- **Declaration and Instantiation**
- **Two-Dimensional Array:** In the **form of matrix** which represents collection of **rows and columns.**

**Syntax:**

```
DataType[][] Array_Name = new int[Row_Size][Column_Size];

int bookCount[][] = new int[3][3];
```

- **Three Dimensional Array:** In the **form of table** and each table contains number of rows and columns.

**Syntax:**

```
DataType[][][] Array_Name = new int[Table_Size][Row_Size][Column_Size];

int bookCount[][][] = new int[3][3][3];
```

# Multi Dimensional Array

**Need for Multi-dimensional Array:**

- Data is stored in the form of **table or matrix form**. It is used to represents the data **in different dimension like row and column wise**.
- It is used to represents **Graph and database** like data structure.
- Specifically used to **find minimum spanning tree** and **connectivity checking between nodes.**

# Multi Dimensional Array

```
/**
 * The ArrayDemo3 class implements an application that illustrate the access of multidimensional
   array elements */
Class ArrayDemo3{
    public static void main(String[] args){
                    int [][] x = new int[][] {{1,2},{3,4},{5,6}}; // initialize values
            for(int i=0; i < x.length; i++) {
                    // print array elements
                        for (int j=0; j < x[i].length; j++) {
                                System.out.print(x[i][j]);
                        }
                    System.out.println();
                }
      }
}
```

**Output**:

1 2
3 4
5 6

# Multi Dimensional Array

```
/**
  * The MovieSeat class implements an application that illustrate the access of multidimensional
    array elements */
Class MovieSeat{
    public static void main(String[] args){
                    String [][] seatType = new String[][] {{1,2},{3,4},{5,6}}; // initialize values
            for(int i=0; i < x.length; i++) {
                     // print array elements
                        for (int j=0; j < x[i].length; j++) {
                                System.out.print(x[i][j]);
                         }
                        System.out.println();
                    }
      }
 }
```

**Output**:

```
1 2
3 4
5 6
```

# Multi Dimensional Array

```
/**
  * The MovieSeat class implements an application that illustrate the access of multidimensional array elements */
public class MovieSeat {
public static void main(String[] args){
        String [][] seatType = new String[][] {{"B","B","A","A","A"},{"A","A","A","B","B"},{"A","B","B","B","B"},{"B","A","A","B","A"}};
        int vipcount = 0, premiumcount = 0, regularcount = 0, viptotal = 5, premiumtotal = 10, regulartotal = 5;
        System.out.println("--MOVIE SEAT ARRANGEMENT--");
        for(int i=0; i < seatType.length; i++) {
        if (i==0)
        System.out.println("--------VIP SEATS--------");
        else if(i==1)
        System.out.println("------PREMIUM SEATS------");
        else if(i==3)
        System.out.println("------REGULAR SEATS------");
```

# Multi Dimensional Array

```
        for (int j=0; j < seatType[i].length; j++) {
            System.out.print(" "+seatType[i][j]+"   ");
            if(i==0 && seatType[i][j].equalsIgnoreCase("B"))
                        vipcount++;
            else if(i>0 && i<3 && seatType[i][j].equalsIgnoreCase("B"))
                        premiumcount++;
            else if(i==3 && seatType[i][j].equalsIgnoreCase("B"))
                        regularcount++;
            }
            System.out.println();
    }
    System.out.println("----SEAT BOOKED DETAIL----");
    System.out.println("--------VIP SEATS--------");
    System.out.println("BOOKED : "+vipcount+" AVAILABLE : "+(viptotal-vipcount)+" TOTAL : "+viptotal);
```

# Multi Dimensional Array

```
        System.out.println("-----PREMIUM SEATS-----");
        System.out.println("BOOKED : "+premiumcount+" AVAILABLE : "+(premiumtotal-
premiumcount)+" TOTAL : "+premiumtotal);
        System.out.println("-----REGULAR SEATS-----");
        System.out.println("BOOKED : "+regularcount+" AVAILABLE : "+(regulartotal-
regularcount)+" TOTAL : "+regulartotal);
    }
}
```

**Output**:

```
--MOVIE SEAT ARRANGEMENT--
--------VIP SEATS--------
 B    B    A    A    A
------PREMIUM SEATS------
 A    A    A    B    B
 A    B    B    B    B
------REGULAR SEATS------
 B    A    A    B    A
----SEAT BOOKED DETAIL----
--------VIP SEATS--------
BOOKED : 2 AVAILABLE : 3 TOTAL : 5
-----PREMIUM SEATS-----
BOOKED : 6 AVAILABLE : 4 TOTAL : 10
-----REGULAR SEATS-----
BOOKED : 2 AVAILABLE : 3 TOTAL : 5
```

# Quiz


1. Which of the following is FALSE about Java array?

a) A java array is always an object

b) Length of array can be changed after the creation of array

c) Arrays in Java are always allocated on heap

d) Array is the example for Non-Primitive type

b) Length of array can be changed after the creation of array

# Quiz


**2. In java array supports,**

a) Primitive type only

b) Object type only

c) Both

d) None of these above

C) Both

# Quiz


**3. Java supports variable size column in multidimensional array**

a) Yes

b) No

b) Yes

# Functions

# Contents

- Introduction
- Function elements
- Quiz

# Introduction

- A **function/Method** is a **block of code** which perform **specific task** and the task is executed when the function is invoked or called.
- Primary **uses of functions**:
  - It allows code r**eusability** (define once and use multiple times)
  - You can **break a complex program** into smaller chunks of code
  - **Reducing duplication** of code
  - Make program shorter and **increases code readability**

# Types

- There are **two types** of functions:
  - **Built-in function:** Java has several functions that are **readily available for use**. In Java, every built-in function should be **part of some class**.
  - **User defined function:** Function created **by user** based on the **need of application**.

- With methods (functions), there are **2 major points** of view
  - **Builder of the method** - responsible for creating the *declaration* and the *definition* of the method (i.e. how it works)
  - **Caller** - somebody (i.e. some portion of code) that *uses* the method to perform a task

# Function Elements

- There are **two** important **elements** of function**:**

1. **Function Definition**

   – It define the **operation of a function.** It consists of the **function signature followed by the function body**. The function prototype includes **function name, parameter list and return type**.

2. **Function Call**

   – It means **call or invoke** the specific function. When a function is invoked, the program **control jumps** to the **called function** to execute the statements that are in the part of that function. Once the called function is executed the program **control passes back to the calling function.**

# Function Elements

- **Function Definition**

**Syntax**:

```
modifier returnType methodName(parameterlists) //
Function signature
{
    //Function Body
}
```

- **Modifiers / Specifiers:** It defines the **visibility of the method** i.e. from where it can be accessible in the application.

# Function Elements

•In Java, there **4 type of the access specifiers**.

- **public:** accessible in all class in your application.
- **protected:** accessible within the class in which it is defined and, in its subclass,(es).
- **private:** accessible only within the class in which it is defined.
- **default (declared/defined without using any modifier) :** accessible within same class and package within which its class is defined.

# Function Elements

- **Function Definition**
  - **The return type :** The data type of the value returned by the method or void if does not return a value.
  - **Method Name :** Name of the function should specify using valid identifier
  - **Java Naming Convention**: It is a **single word** that should be a verb in lowercase or **multi-word**, that begins with a verb in lowercase followed by adjective or noun. After the first word, **first letter of each word should be capitalized. Example**: computeAddition, setName
  - **Parameter list :** Comma separated list of the **input parameters** are defined, preceded with their data type, within the enclosed parenthesis. If there are no parameters, you must use **empty parentheses ().**
  - **Method body :** It is enclosed between braces. The code specifies the **task to be done**.

# Function Elements

- **Function Call**
  - **Static method** is invoked by the **class name** or **without using object**. (**Refer Example**)
  - **Example**: className.methodName(argumentList)
  - **Non static method** is invoked by the **instance/object** of the class (**We Will discuss later**)
  - **Example:** objectName.methodName(argumentList)

# Return Values

- **Function return types**: Function return values of **any valid types**. It must **return data** that **matches their return type.**
- **Example:**

```
public int addTwoInt(int a, int b){
        return a+b;   // return integer
 }
```

```
 //void method return nothing
public void printName(String name){
   System.out.println("Hello World!!!"); // return
void
 }
```

# Arguments and Parameters

- **Arguments and Parameters:** The terms parameter and argument can be used for the same thing information that are passed into a function.

  – **Arguments** –An argument is a value that is passed during a **function call or calling function**.

  – **Parameters** – Parameters used in the function definition or called function. A parameter is a variable defined in the **function definition or calling function**.

**Note:**

- During program execution, the **values in the actual arguments** is **assigned to the formal parameter**.

# Function Elements

```
//Called Function

public int addTwoInt(int a, int b){

  return a+b;    //Function Body

}

//Calling Function

public static void main (String[] args){

   int sum;
                        Function Call
   sum=addTwoInt(5,6);

   System.out.println("Sum of two integer values :"+ sum);

}
```

Here,
a & b is formal parameters

Here,
5 & 6 is actual arguments

# Function Elements

```
/**
 * The MethodDeclareDemo class implements an application that
 * Illustrate the user defined methods(static) */
class MethodDeclareDemo{
        //User Defined Method
        public int static addTwoInt(int a, int b){  //a, b is parameters
                return a+b;
         }
        public static void main (String[] args){
              int sum;
              sum=addTwoInt(5,6);  //5, 6 is arguments
               System.out.println("Sum of two integer values :"+ sum);
         }
    }
```

**Output**:

Sum of two integer values :11

# Function Elements

```
/**
 * The FunctionDeclare class implements an application that display the Movie details
 * using the user defined methods(static) */
public class FunctionDeclare {
    static void getMovieDetail(String moviename,String moviedescription,int movieduration,String movielanguage,
    String moviereleasedate,String moviecountry,String moviegenre) {
                System.out.println("Movie Title : "+moviename);
                System.out.println("Movie Description : "+moviedescription);
                System.out.println("Movie Duration : "+movieduration);
                System.out.println("Movie Language : "+movielanguage);
                System.out.println("Movie Release Date : "+moviereleasedate);
                System.out.println("Movie Country : "+moviecountry);
                System.out.println("Movie Genre : "+moviegenre);
    }
```

# Function Elements

```
    public static void main(String[] args) {
        String moviename = "AAA";
        String moviedescription = "Dramaof1945";
        int movieduration = 3;
        String movielanguage = "English";
        String moviereleasedate = "25/03/2022";
        String moviecountry = "XYZ";
        String moviegenre = "THRILLER";
       System.out.println("-------Movie Detail-------");
       getMovieDetail(moviename, moviedescription, movieduration, movielanguage, moviereleasedate,
moviecountry, moviegenre);
        System.out.println("--------------------------");
    }   }
```

# Function Elements

**Output**:

-------Movie Detail-------
Movie Title : AAA
Movie Description : Dramaof1945
Movie Duration : 3
Movie Language : English
Movie Release Date : 25/03/2022
Movie Country : XYZ
Movie Genre : THRILLER
--------------------------

# Quiz


**1. Java method signature is a combination of ___.**

a)returnType

b) methodName

c) Argumentlist

d) All the above

d) All the above

# Quiz


**2. What the naming convention should be for Methods in Java?**

a) It should start with lowercase letter.

b) If name contains many words then first word's letter start with lowercase only and other words will start with uppercase

c) It should be a verb such as show(), count(), describe()

d) All the above

d) All the above

# Quiz


**3. Which is not an advantage of the function?**

a) Re-usability

b) Makes problem simple to solve

c) Efficiency

d) Easy to Understand the Solution

c) Efficiency

# A class, in JAVA , is where we teach objects how to behave.