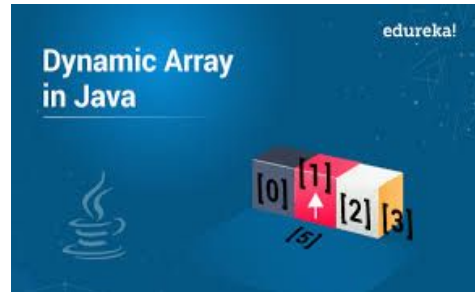
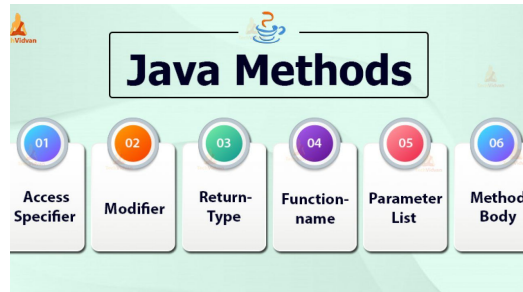


CHAPTER 3

METHODS AND ARRAYS



SUBJECT: OOP-I
CODE: 3140705

PREPARED BY:
ASST. PROF. NENSI KANSAGARA
(CSE DEPARTMENT, ACET)

DEFINING AND CALLING METHODS

In mathematics, we might have studied about functions. For example, $f(x) = x^2$ is a function that returns a squared value of x .

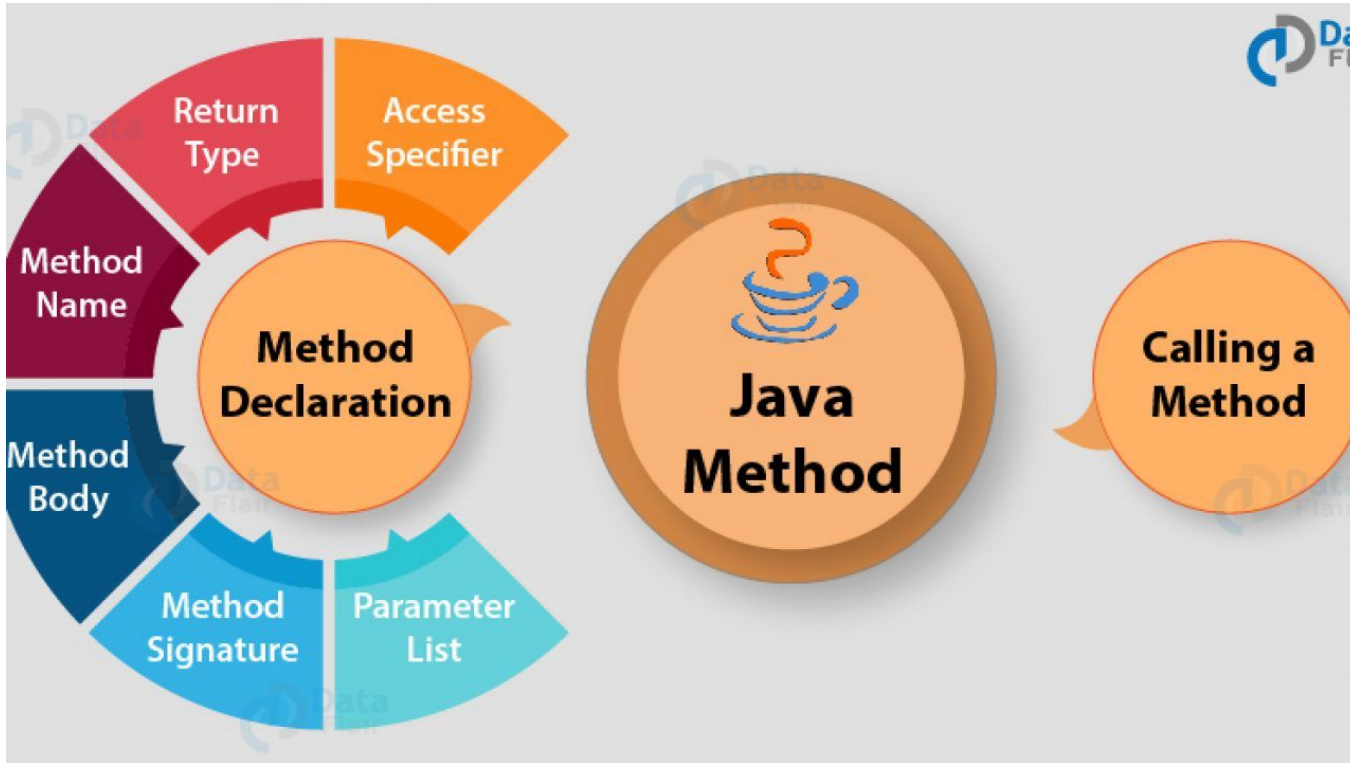
If $x = 2$, then $f(2) = 4$

If $x = 3$, $f(3) = 9$

and so on.

Similarly, in computer programming, a function is a block of code that performs a specific task.

In object-oriented programming, the method is a jargon used for function. Methods are bound to a class and they define the behavior of a class.



TYPES OF JAVA METHODS

→ Depending on whether a method is defined by the user, or available in the standard library, there are two types of methods in Java:

- ◆ Standard Library Methods
- ◆ User-defined Methods

STANDARD LIBRARY METHOD

- The standard library methods are built-in methods in Java that are readily available for use. These standard libraries come along with the Java Class Library (JCL) in a Java archive (*.jar) file with JVM and JRE.
- For example,
 - ◆ `print()` is a method of `java.io.PrintSteam`. The `print("...")` method prints the string inside quotation marks.
 - ◆ `sqrt()` is a method of `Math` class. It returns the square root of a number.

```
public class Main {  
    public static void main(String[] args) {  
  
        // using the sqrt() method  
        System.out.print("Square root of 4 is: " + Math.sqrt(4));  
    }  
}
```

Output:

Square root of 4 is: 2.0

USER DEFINE METHOD

We can also create methods of our own choice to perform some task. Such methods are called user-defined methods.

How to create a user-defined method?

Here is how we can create a method in Java:

```
public static void myMethod() {  
  
    System.out.println("My Function called");  
  
}
```

- Here, we have created a method named `myMethod()`. We can see that we have used
- — the `public`, `static` and `void` before the method name.
- ◆ `public` - access modifier. It means the method can be accessed from anywhere. To learn more, visit [Java access modifier](#)
 - ◆ `static` - It means that the method can be accessed without any objects. To learn more, visit the [Java static Keyword](#).
 - ◆ `void` - It means that the method does not return any value. We will learn more about this later in this tutorial.

This is a simple example of how we can create a method. However, the complete syntax of a method definition in Java is:

```
modifier static returnType nameOfMethod (parameters) {  
  
    // method body  
  
}
```

Here,

- modifier - It defines access types whether the method is public, private and so on.
- static - If we use the `static` keyword, it can be accessed without creating objects.

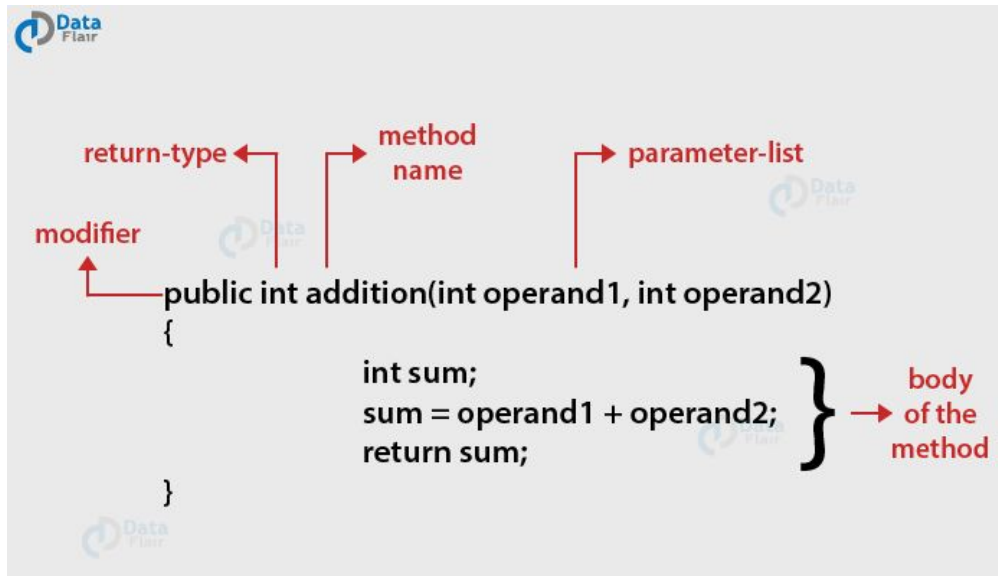
For example, the `sqrt()` method of standard `Math class` is static. Hence, we can directly call `Math.sqrt()` without creating an instance of `Math class`.

- returnType - It specifies what type of value a method returns For example if a method has `int` return type then it returns an integer value.

→ A method can return native data types (int, float, double, etc), native objects (String, Map, List, etc), or any other built-in and user-defined objects.

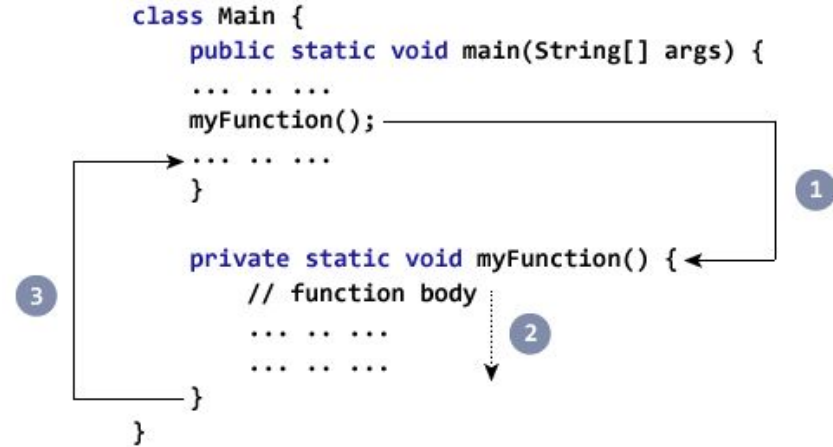
— — —
If the method does not return a value, its return type is void.

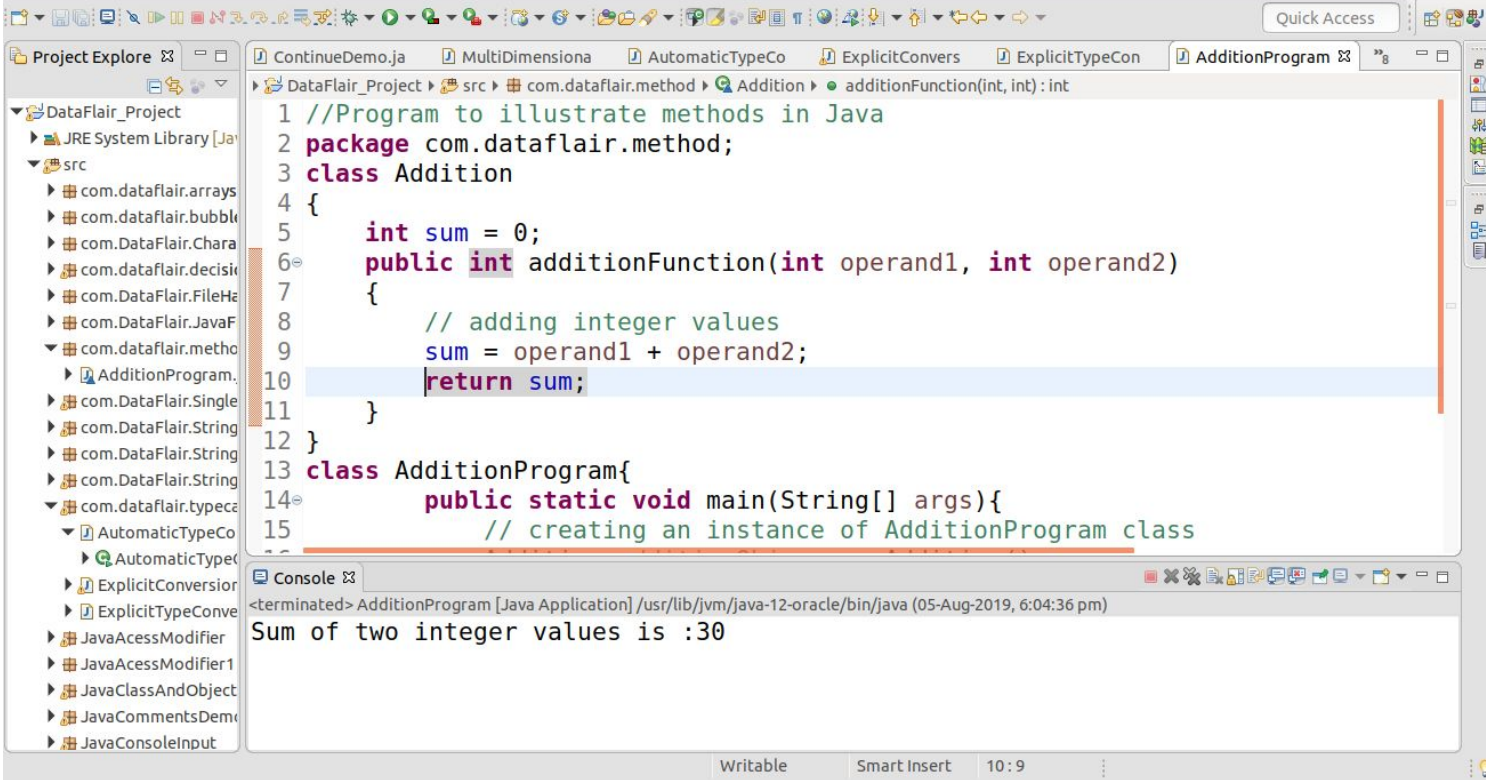
→ nameOfMethod - It is an **identifier** that is used to refer to the particular method in a program.



HOW TO CALL JAVA METHOD

1. While executing the program code, it encounters `myFunction();` in the code.
2. The execution then branches to the `myFunction()` method and executes code inside the body of the method.
3. After the execution of the method body, the program returns to the original state and executes the next statement after the method call.





The screenshot displays the Eclipse IDE interface. The Project Explorer on the left shows the project structure: DataFlair_Project > src > com.dataflair.method > AdditionProgram.java. The main editor window shows the following Java code:

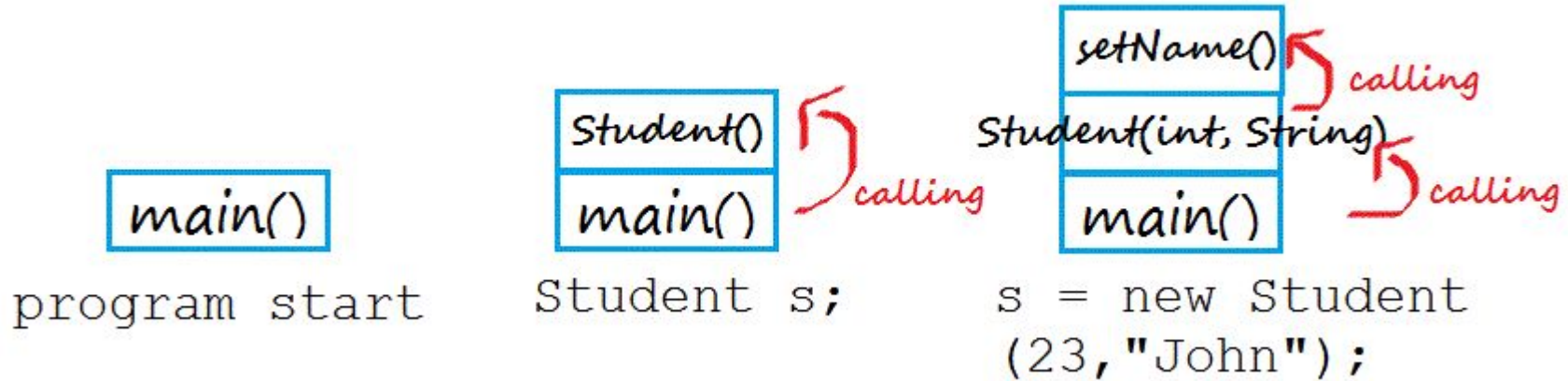
```
1 //Program to illustrate methods in Java
2 package com.dataflair.method;
3 class Addition
4 {
5     int sum = 0;
6     public int additionFunction(int operand1, int operand2)
7     {
8         // adding integer values
9         sum = operand1 + operand2;
10        return sum;
11    }
12 }
13 class AdditionProgram{
14     public static void main(String[] args){
15         // creating an instance of AdditionProgram class
```

The console window at the bottom shows the output of the program:

```
<terminated> AdditionProgram [Java Application] /usr/lib/jvm/java-12-oracle/bin/java (05-Aug-2019, 6:04:36 pm)
Sum of two integer values is :30
```

The status bar at the bottom indicates 'Writable', 'Smart Insert', and '10 : 9'.

CALL STACK



PASSING ARGUMENTS BY VALUES

Passing Parameters by Value means calling a method with a parameter. Through this, the argument value is passed to the parameter.

```
public class SwappingExample {  
    public static void swapFunction(int a, int b) {  
        int c = a+b;  
        System.out.println("Sum of the given numbers is ::"+c);  
    }  
  
    public static void main(String[] args) {  
        int a = 30;  
        int b = 45;  
        swapFunction(a, b);  
    }  
}
```

Output

```
Sum of the given numbers is ::75
```

OVERLOADING METHODS

- In Java, two or more **methods** can have same name if they differ in parameters (different number of parameters, different types of parameters, or both). These methods are called overloaded methods and this feature is called method overloading. For example:
- Here, the `func()` method is overloaded. These methods have the same name but accept different arguments.
- Notice that, the return type of these methods is not the same. Overloaded methods may or may not have different return types, but they must differ in parameters they accept.

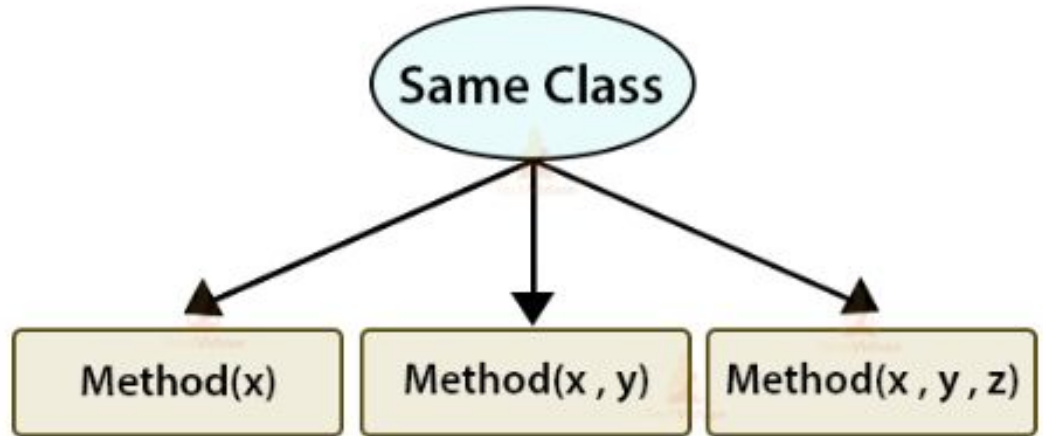
→ Two or more methods can have same name inside the same class if they accept different arguments. This feature is known as method overloading.

→ Method overloading is achieved by either:

- ◆ changing the number of arguments.
- ◆ or changing the datatype of arguments.

→ Method overloading is not possible by changing the return type of methods.

```
void func() { ... }  
void func(int a) { ... }  
float func(double a) { ... }  
float func(int a, float b) { ... }
```



OVERLOADING

```
class Cat{  
  
public void Sound(){  
System.out.println("meow");  
}  
  
//overloading method  
public void Sound(int num){  
    for(int i=0; i<2;i++){  
        System.out.println("meow");  
    }  
}  
}
```

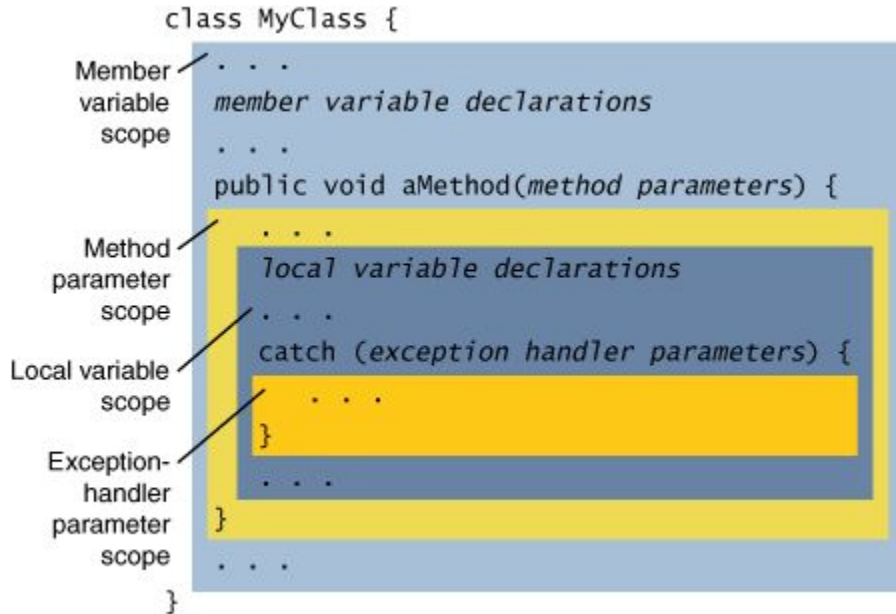
Same method
name but
different
parameters

AMBIGUITY IN METHOD OVERLOADING

```
Test.java ✖
1 package com.journaldev.errors;
2
3 public class Test {
4     public void foo(Object o) {
5         System.out.println("Object");
6     }
7
8     public void foo(String s) {
9         System.out.println("String");
10    }
11
12    public void foo(Integer i) {
13        System.out.println("Integer");
14    }
15
16    public static void main(String[] args) {
17        The method foo(Object) is ambiguous for the type Test
18    }
19 }
```

You will get compile time **error** as The **method** `foo(Object)` is **ambiguous** for the type `Test` because both `String` and `Integer` class have `Object` as parent class and there is no inheritance. So **java** compiler doesn't consider any of them to be more specific, hence the **method ambiguous call error**.

SCOPE OF VARIABLE



- Scope of a variable is the part of the program where the variable is accessible. Like C/C++, in Java, all identifiers are lexically (or statically) scoped, i.e. scope of a variable can be determined at compile time and independent of function call stack.
- Java programs are organized in the form of classes. Every class is part of some package. Java scope rules can be covered under following categories.

```
public class Scope2
{
    public static void main (String[] args )
    {
        {
            double r;
            r = 3.14;
        }
        System.out.println(r);
    }
}
```

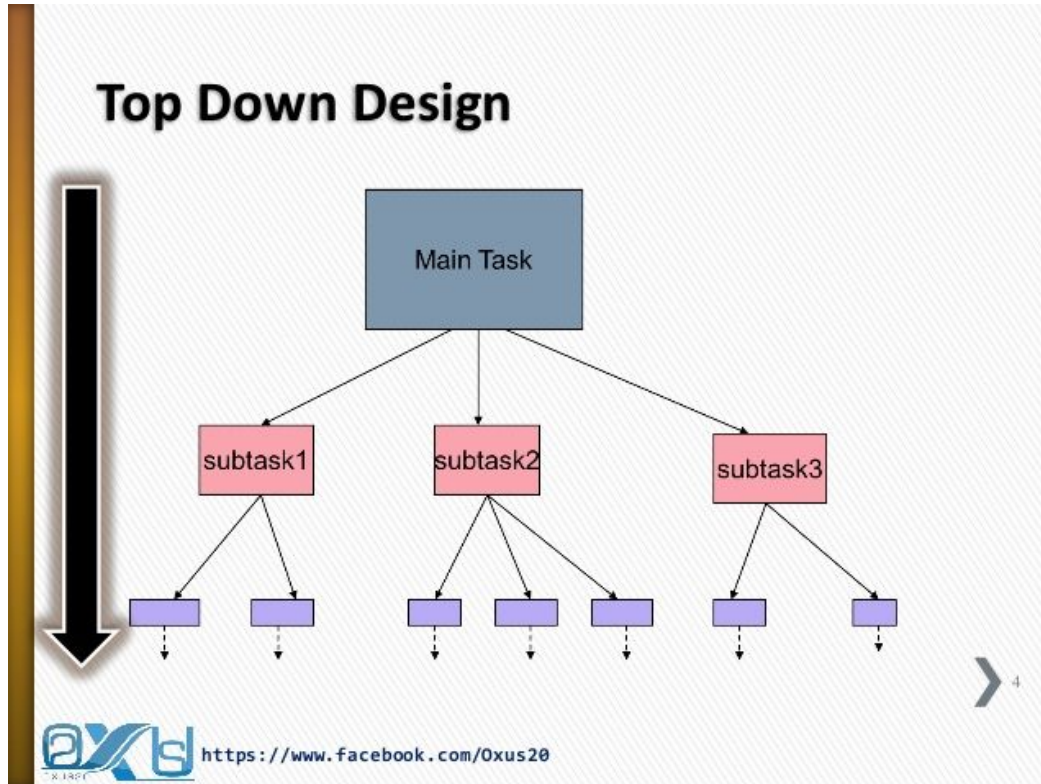
Scope of the variable r

*use variable OUTSIDE its scope
Error !!*

ABSTRACTION AND STEPWISE REFINEMENT

Abstract method in **Java** with examples. A **method** without body (no implementation) is known as **abstract method**. A **method** must always be declared in an **abstract** class, or in other words you can say that if a class has an **abstract method**, it should be declared **abstract** as well.

TOP DOWN DESIGN



TOP DOWN AND BOTTOM UP IMPLEMENTATION

- **Stepwise refinement** refers to the progressive **refinement** in small steps of a program specification into a program. Sometimes, it is called top-down design. ... Wirth said, "It is here considered as a sequence of design decisions concerning the decomposition of tasks into subtasks and of data into data structures."
- Stepwise refinement: design a problem solution by
 - ◆ stating the solution at a high level
 - ◆ refining steps of the solution into simpler steps
 - ◆ repeating step 2, until steps are simple enough to execute
- Decompose based on function of each step
- Makes heavy use of pseudocode

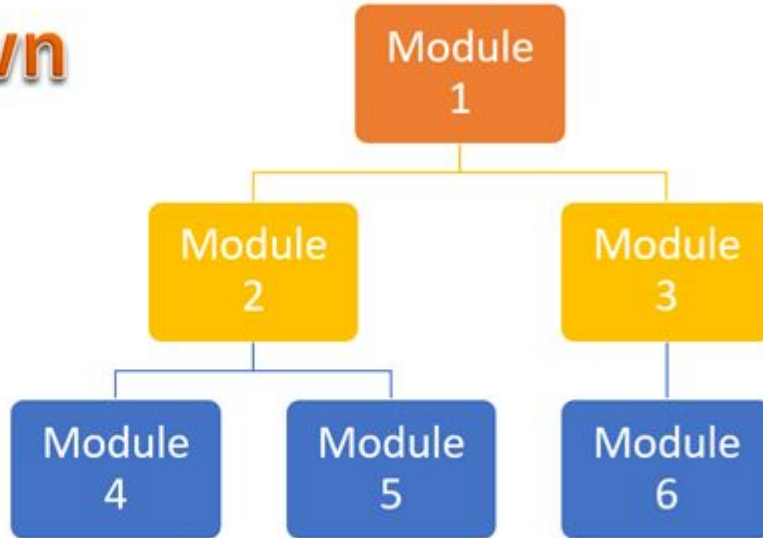
Top Down

↓

Hybrid

↑

Bottom Up



ARRAY USING JAVA

SINGLE DIMENSIONAL ARRAY

-
- An array is a collection of similar types of data. It is a container that holds data (values) of one single type. For example, you can create an array that can hold 100 values of int type.
 - In Java, arrays are a fundamental construct that allows you to store and access a large number of values conveniently.

HOW TO DECLARE AN ARRAY?

```
dataType[] arrayName;
```

- dataType - it can be **primitive data types** like int, char, double, byte, etc. or **Java objects**
- arrayName - it is an **identifier**
- Let's take an example,

```
double[] data;
```

Here, data is an array that can hold values of type double.

But, how many elements can array this hold?

Good question! We have to allocate memory for the array. The memory will define the number of elements that the array can hold.

```
data = new Double[10];
```

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



Array age of length 5

Java Array Index

General Form of Java Array Initialization

The **type** determines what type of data the array will hold

new is a special operator that allocates memory.

type array-var = new type[size];

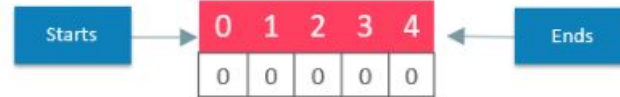
array-var is the array variable that is linked to the array.

size specifies the number of elements in the array

1

data type size of array
↓ ↓
`int[] a = new int[5];`

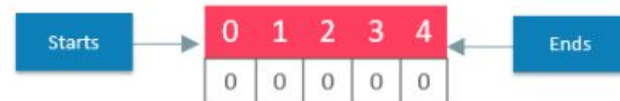
Index has to be given in square brackets



2

data type size of array
↓ ↓
`int a[] = new int[5];`

Index has to be given in square brackets



```
1 class MyArray{
2
3 public static void main(String args[]){
4
5 int month_days[ ] = {31,28,31,30,31,30,31,30,31,30,31};
6
7 System.out.println("April has " + month_days[3] + "days.");
8
9 }
10
11 }
```

```
1 public static void main(String args[]) {
2     int month_days[];
3     month_days = new int[12];
4     month_days[0] = 31;
5     month_days[1] = 28;
6     month_days[2] = 31;
7     month_days[3] = 30;
8     month_days[4] = 31;
9     month_days[5] = 30;
10    month_days[6] = 31;
11    month_days[8] = 30;
12    month_days[9] = 31;
13    month_days[10] = 30;
14    month_days[11] = 31;
15    System.out.println("April has " + month_days[3] + " days.");
16 }
17 }
```

PASSING AN ARRAY TO METHOD

- Just as you can pass primitive type values to methods, you can also pass arrays to a method. To pass an array to a method, specify the name of the array without any square brackets within the method call. Unlike C/C++, in Java every array object knows its own length using its length field, therefore while passing array's object reference into a method, we do not need to pass the array length as an additional argument

```

class sortNumbers
{
    public static void main(String[] args)
    {
        int[] data={40,50,10,30,20,5};
        System.out.println("Unsorted List is :");
        display(data);
        sort(data);
        System.out.println("\nSorted List is :");
        display(data);
    }
    static void display(int num[])
    {
        for(int i=0; i<num.length;i++)
            System.out.print(num[i] + " ");
    }
    static void sort(int num[])
    {
        int i, j, temp;
        for(i=0; i<num.length-i;i++)
        {
            for(j=0; j<num.length-i-1;j++)
            {
                if(num[j]>num[j+1])
                {
                    temp = num[j];
                    num[j] = num[j+1];
                    num[j+1] = temp;
                }
            }
        }
    }
}

```

```

Command Prompt
F:\Java>javac sortNumbers.java
F:\Java>java sortNumbers
Unsorted List is :
40 50 10 30 20 5
Sorted List is :
10 20 5 30 40 50
F:\Java>_

```


RETURNING ARRAY FROM METHOD

```
import java.util.Arrays;
import java.util.Scanner;

public class ReturningAnArray {
    public int[] createArray() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the array that is to be created:: ");
        int size = sc.nextInt();
        int[] myArray = new int[size];
        System.out.println("Enter the elements of the array ::");

        for(int i=0; i<size; i++) {
            myArray[i] = sc.nextInt();
        }
        return myArray;
    }

    public static void main(String args[]) {
        ReturningAnArray obj = new ReturningAnArray();
        int arr[] = obj.createArray();
        System.out.println("Array created is :: "+Arrays.toString(arr));
    }
}
```

Output

```
Enter the size of the array that is to be created::
5
Enter the elements of the array ::
23
47
46
58
10
Array created is :: [23, 47, 46, 58, 10]
```

TWO DIMENSIONAL ARRAY AND ITS PROCESSING

Multidimensional arrays are *arrays of arrays*.

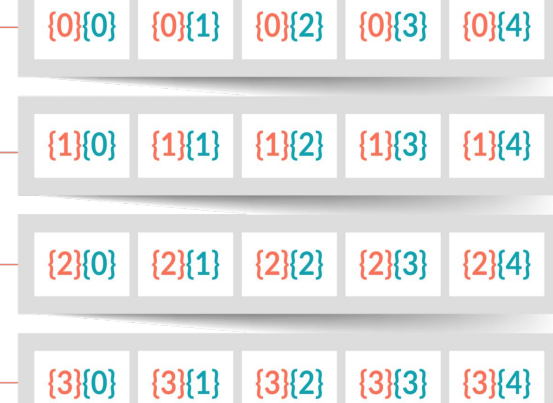
This allocates a 4 by 5 array and assigns it to **Mul**.

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

edureka!

Left index determines row.

Right index determines column.



```
1 class Mul2D{
2     public static void main(String args[]) {
3         int mul2d[][]= new int[4][5];
4         int i, j, k = 0;
5         for(i=0; i<4; i++)
6             for(j=0; j<5; j++) {
7                 Mul2D[i][j] = k;
8                 k++;
9             }
10        for(i=0; i<4; i++) {
11            for(j=0; j<5; j++);
12            System.out.print(mul2d[i][j] + " ");
13            System.out.println();
14        }
15    }
16 }
```

This program generates the following output:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

These are other Multidimensional arrays representation of other data types.

```
int [][]a= new int [2][2];
```

	0	1
0	1	4
1	4	5

2 x 2 dimensional int array

```
char [][]a= new char[3][2];
```

	0	1
0	s	a
1	g	v
2	v	d

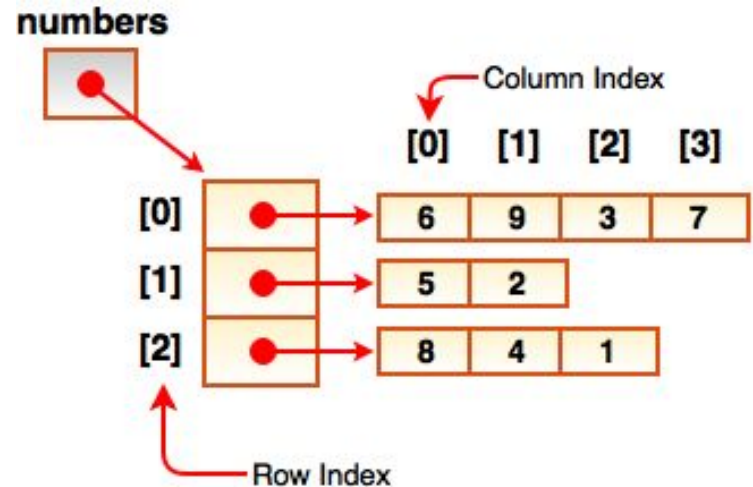
```
float [][]a= new float[5][5];
```

	0	1	2	3	4
0	2.2	3.4	5.0	3.3	1.2
1	7.8	9.0	1.1	2.9	5.5
2	2.0	3.0	7.8	9.8	9.9
3	5.7	6.6	8.8	5.3	2.7
4	1.8	4.4	7.6	1.0	1.1

5 x 5 dimensional float array

CONCEPT OF RAGGED ARRAY

→ "Ragged Array" is an "Array of Array". This means one can create an **array** in such a way that each element of the **array** is a reference to another **array** of same type. For example, a two-dimensional **array** of 3 x 2 refers three rows and two columns (i.e.) each row has two columns.



```
class Main
{
    public static void main(String[] args)
    {
        // Declaring 2-D array with 2 rows
        int arr[][] = new int[2][];

        // Making the above array Jagged

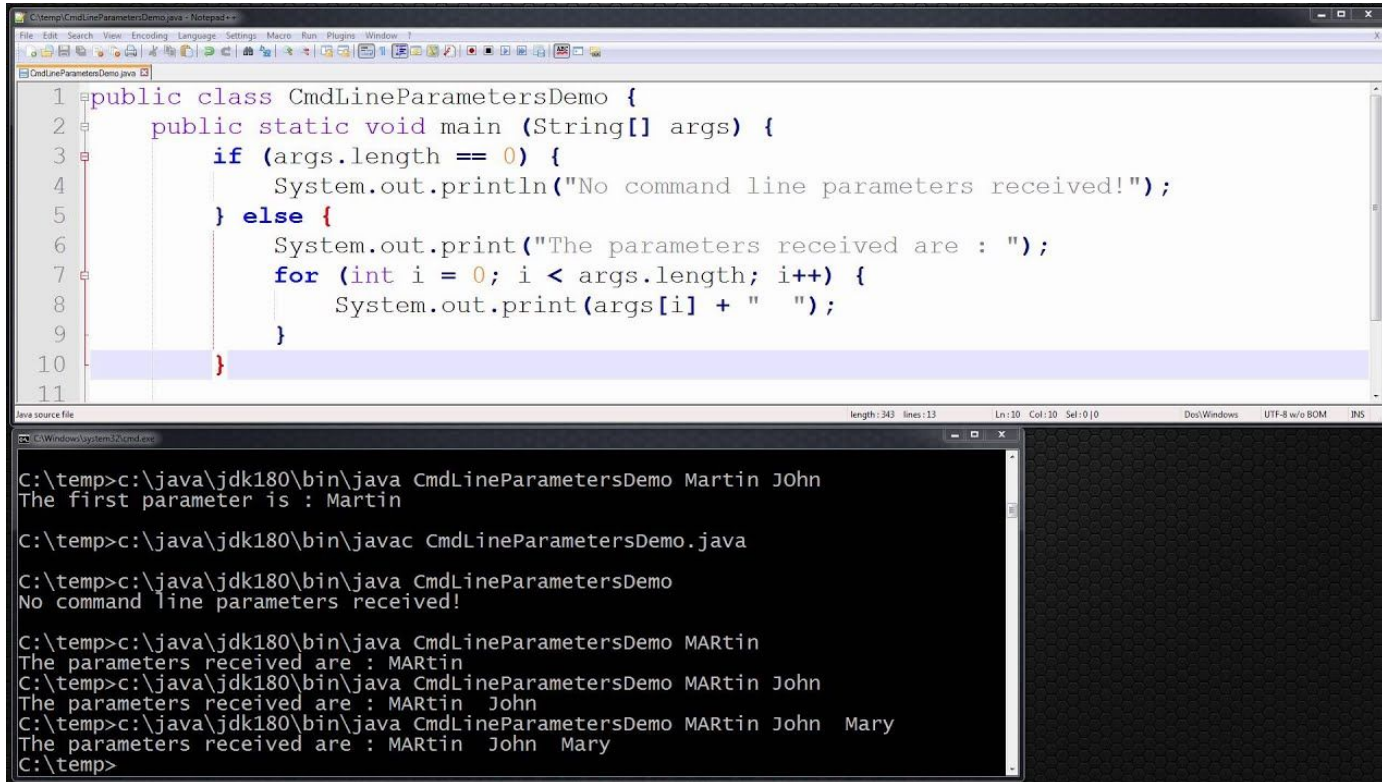
        // First row has 3 columns
        arr[0] = new int[3];

        // Second row has 2 columns
        arr[1] = new int[2];

        // Initializing array
        int count = 0;
        for (int i=0; i<arr.length; i++)
            for(int j=0; j<arr[i].length; j++)
                arr[i][j] = count++;

        // Displaying the values of 2D Jagged array
        System.out.println("Contents of 2D Jagged Array");
        for (int i=0; i<arr.length; i++)
        {
            for (int j=0; j<arr[i].length; j++)
                System.out.print(arr[i][j] + " ");
            System.out.println();
        }
    }
}
```

COMMAND LINE ARGUMENTS



```
1 public class CmdLineParametersDemo {
2     public static void main (String[] args) {
3         if (args.length == 0) {
4             System.out.println("No command line parameters received!");
5         } else {
6             System.out.print("The parameters received are : ");
7             for (int i = 0; i < args.length; i++) {
8                 System.out.print(args[i] + " ");
9             }
10        }
11    }
```

```
C:\temp>c:\java\jdk180\bin\java CmdLineParametersDemo Martin John
The first parameter is : Martin

C:\temp>c:\java\jdk180\bin\javac CmdLineParametersDemo.java

C:\temp>c:\java\jdk180\bin\java CmdLineParametersDemo
No command line parameters received!

C:\temp>c:\java\jdk180\bin\java CmdLineParametersDemo MARTin
The parameters received are : MARTin
C:\temp>c:\java\jdk180\bin\java CmdLineParametersDemo MARTin John
The parameters received are : MARTin John
C:\temp>c:\java\jdk180\bin\java CmdLineParametersDemo MARTin John Mary
The parameters received are : MARTin John Mary
C:\temp>
```

PASSING TWO DIMENSIONAL ARRAY TO METHODS

- Just like one-dimensional arrays, a two-dimensional array can also be passed to a method and it can also be returned from the method. The syntax is similar to one-dimensional arrays with an exception that an additional pair of square brackets is used.


```

public class Transpose
{
    public static void main(String[] args)
    {
        int[] [] table= {{5,6,7},{3,4,2}};
        int[] [] result;
        System.out.println("Matrix Before Transpose : ");
        for(int i=0;i<table.length;i++)
        {
            for(int j=0;j<table[i].length;j++)
                System.out.print(table[i][j]+" ");
        }
        result=transpose(table);
        System.out.println("\nMatrix After Transpose :");
        for(int i=0;i<result.length;i++)
        {
            for(int j=0;j<result[i].length;j++)
                System.out.print(result[i][j]+" ");
            System.out.println();
        }
    }
    static int[][] transpose(int[][] a)
    {
        int[][] temp=new int[a[0].length][a.length];
        for(int i=0;i<a[0].length;i++)
        {
            for(int j=0;j<a.length;j++)
                temp[i][j]= a[j][i];
        }
        return temp;
    }
}

```

```

Command Prompt
P:\Java>java Transpose
Matrix Before Transpose :
5 6 7 3 4 2
Matrix After Transpose :
5 3
6 4
7 2
P:\Java>

```

MULTIDIMENSIONAL ARRAY

→ A multidimensional array is an array of arrays. Each element of a multidimensional array is an array itself.

For example,

→ `int[][] a = new int[3][4];`

→ Here, we have created a multidimensional array named `a`. It is a 2-dimensional array, that can hold a maximum of 12 elements,

	Column 1	Column 2	Column 3	Column 4
Row 1	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>
Row 2	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>
Row 3	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>

How to initialize a 2d array in Java?

Here is how we can initialize a 2-dimensional array in Java.

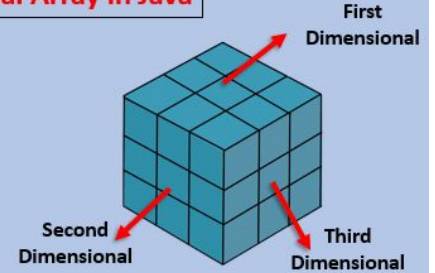
```
int[][] a = {  
    {1, 2, 3},  
    {4, 5, 6, 9},  
    {7},  
};
```

Multidimensional Array in Java

Types of Multidimensional Array in Java

	Column 0	Column 1	Column 2
Row 0	X[0][0]	X[0][1]	X[0][2]
Row 1	X[1][0]	X[1][1]	X[1][2]
Row 2	X[2][0]	X[2][1]	X[2][2]

2D-Array



3D-Array

www.educba.com

SEARCHING AND SORTING ARRAYS AND ARRAY CLASS

SEARCHING ARRAY

How to sort an array and search
an element inside it?

LINEAR SEARCH

Linear search is used to search a key element from multiple elements. Linear search is less used today because it is slower than binary search and hashing.

Algorithm:

- Step 1: Traverse the array
- Step 2: Match the key element with array element
- Step 3: If key element is found, return the index position of the array element
- Step 4: If key element is not found, return -1

```
public class HelloWorld {
    public static void main(String[] args) {
        int[] a = { 2, 5, -2, 6, -3, 8, 0, -7, -9, 4 };
        int target = 0;

        for (int i = 0; i < a.length; i++) {
            if (a[i] == target) {
                System.out.println("Element found at index " + i);
                break;
            }
        }
    }
}
```

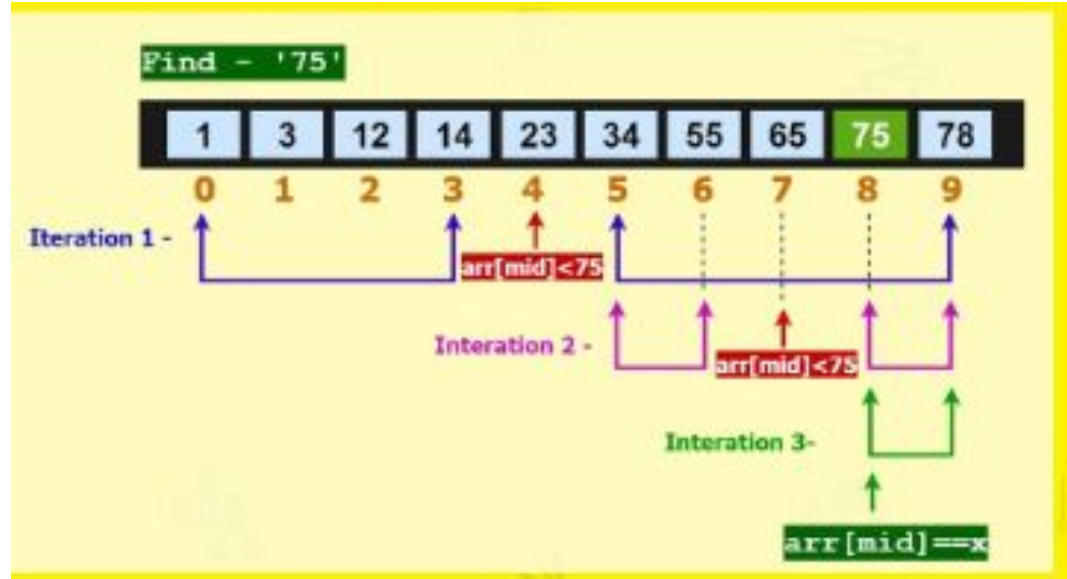
Result

The above code sample will produce the following result.

```
Element found at index 6
```

BINARY SEARCH

-
- Binary search is used to search a key element from multiple elements. Binary search is faster than linear search.
 - In case of binary search, array elements must be in ascending order. If you have unsorted array, you can sort the array using `Arrays.sort(arr)` method.



BINARY SEARCH



Array



Divide and Conquer

Best

Average

Worst

$O(1)$

$O(\log n)$

$O(\log n)$

search (A, t)

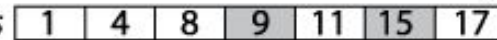
1. low = 0
 2. high = n - 1
 3. **while** (low \leq high) **do**
 4. ix = (low + high)/2
 5. **if** (t = A[ix]) **then**
 6. **return true**
 7. **else if** (t < A[ix]) **then**
 8. high = ix - 1
 9. **else** low = ix + 1
 10. **return false**
- end**

search (A, 11)

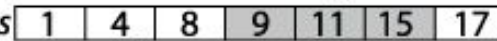
first pass low ix high



second pass low ix high



third pass low ix high



*explored
elements*

```
2 public class BinarySearch
3 {
4     //algorithm for binary search
5     public static int binarySearch(int[] arr, int key, int low, int high){
6         int index = -1;
7         while (low <= high) {
8             int mid = (low + high) / 2;
9             if (arr[mid] < key) {
10                 low = mid + 1;
11             }
12             else if (arr[mid] > key) {
13                 high = mid - 1;
14             }
15             else if (arr[mid] == key) {
16                 index = mid;
17                 break;
18             }
19         }
20         return index;
21     }
22     public static void main(String[] args) {
23         int[] arr = {10,15,20,25,30,35,40};
24         int key = 30;    int low = 0;    int high = arr.length-1;
25         int index = binarySearch(arr,key,low,high);
26         if (index == -1) {
27             System.out.println(key+" is not present");
28         }
29         else
30             System.out.println(key+" is present at the index "+index);
31     }
32 }
```

Binary Search In Java

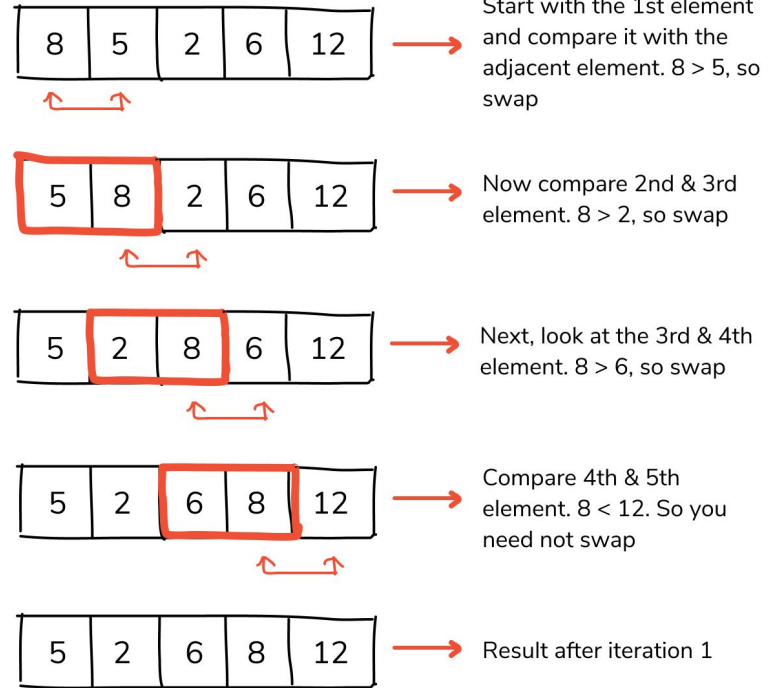
{javacodekorner.blogspot.com}

SORTING ARRAY

How to sort an array and search
an element inside it?

BUBBLE SORT

-
- We can create a java program to sort array elements using bubble sort. Bubble sort algorithm is known as the simplest sorting algorithm.
 - In bubble sort algorithm, array is traversed from first element to last element. Here, current element is compared with the next element. If current element is greater than the next element, it is swapped.



```
public class HelloWorld {
    static void bubbleSort(int[] arr) {
        int n = arr.length;
        int temp = 0;
        for(int i = 0; i < n; i++) {
            for(int j=1; j < (n-i); j++) {
                if(arr[j-1] > arr[j]) {
                    temp = arr[j-1];
                    arr[j-1] = arr[j];
                    arr[j] = temp;
                }
            }
        }
    }
    public static void main(String[] args) {
        int arr[] = { 2, 5, -2, 6, -3, 8, 0, -7, -9, 4 };
        System.out.println("Array Before Bubble Sort");

        for(int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
        bubbleSort(arr);
        System.out.println("Array After Bubble Sort");

        for(int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

Result

The above code sample will produce the following result.

```
Array Before Bubble Sort
2 5 -2 6 -3 8 0 -7 -9 4
Array After Bubble Sort
-9 -7 -3 -2 0 2 4 5 6 8
```

Thank you!

A red ballpoint pen is shown in the process of writing the words "Thank you!" in a black, cursive script on a white surface. The pen is positioned at the end of the word "you", with its tip touching the final exclamation point.