JAVA BASICS

4KCJ001 – LECTURE 2

By
Sudha Agarwal
sudha.agarwal@4kitsolutions.com

http://www.4kitsolutions.com/

INDEX

- Variables
- Methods
- Access Modifiers
- Method Overloading
- Non Access Modifiers
- Final
- Static
- Constructors
- Arrays

JAVA VARIABLES

❖Variable is a name of memory location. There are three types of variables in java: local, instance and static.

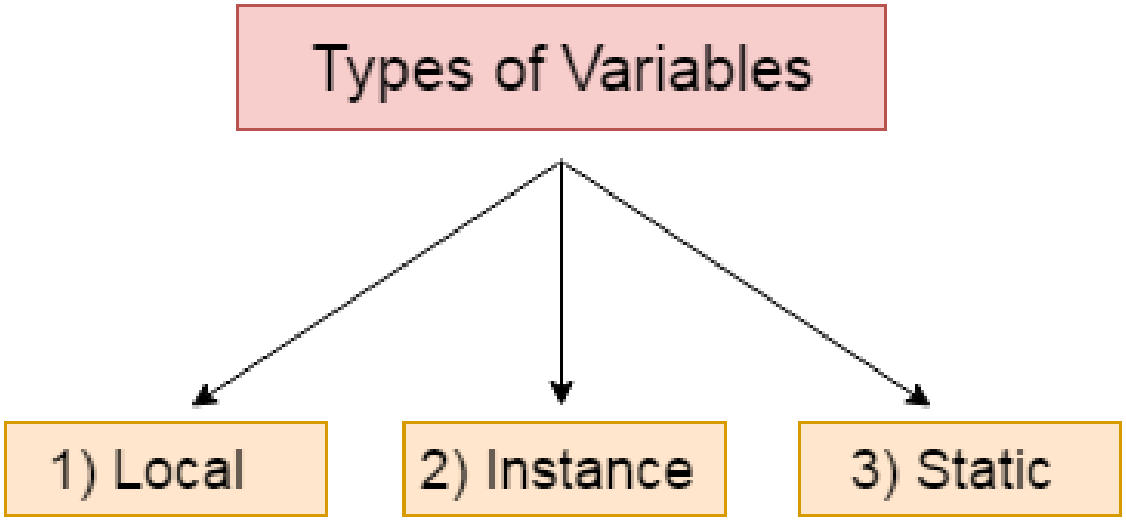
❖There are two types of data types in java: primitive and non-primitive.

❖Variable is name of reserved area allocated in memory. In other words, it is a name of memory location.

Types of Variable

- There are three types of variables in java:

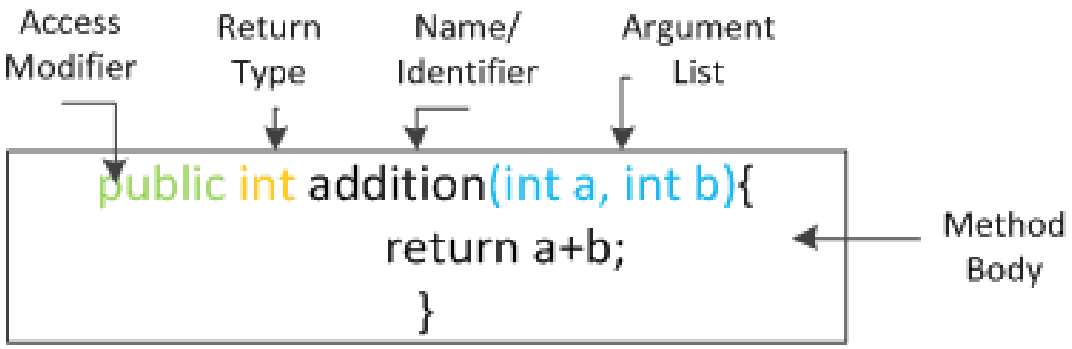
- local variable
- instance variable
- static variable



METHODS

❖A method is a program module that contains a series of statements that carry out a task.

❖To execute a method, you invoke or call it from another method.



ACCESS MODIFIERS

- Each object has members (members can be variable and methods) which can be declared to have specific access.

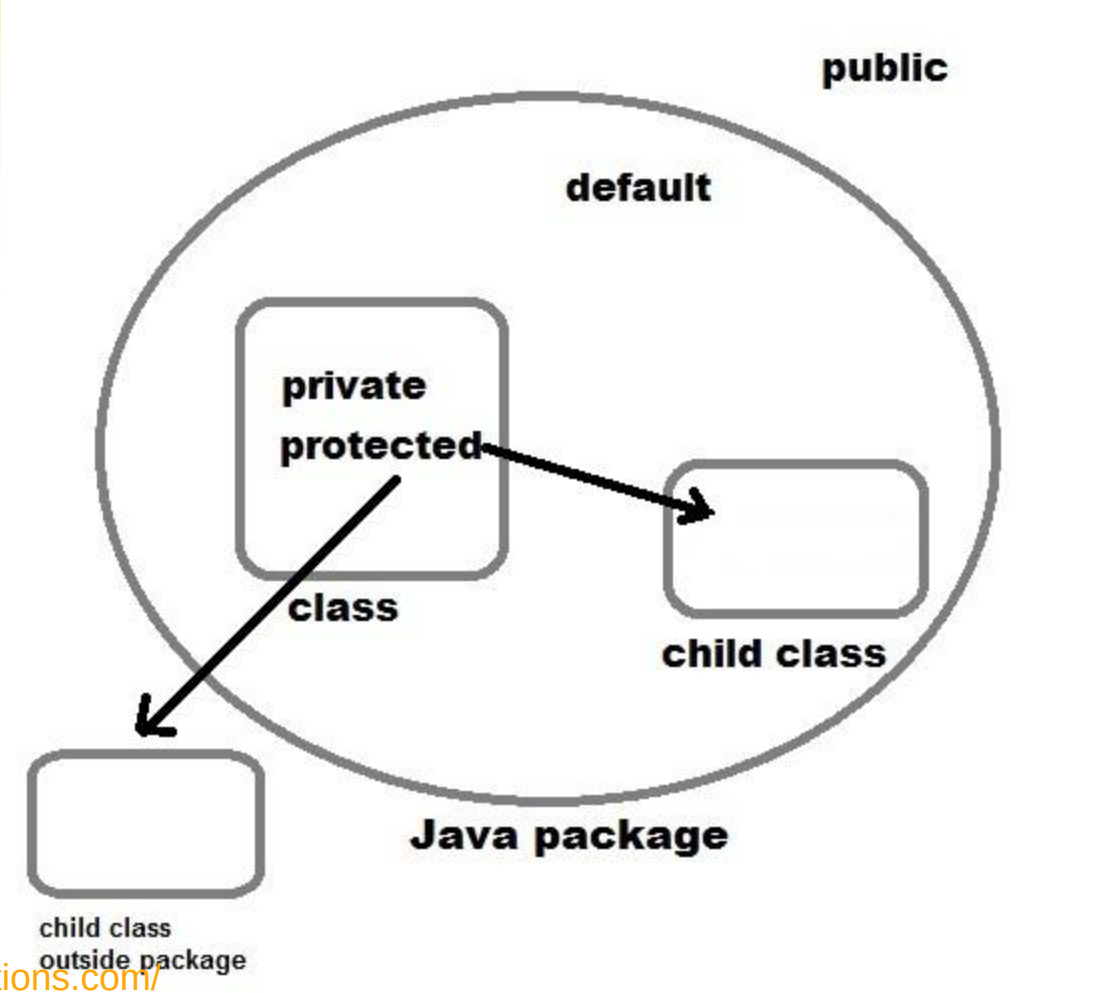
- In Java, modifiers are categorized into two types,
  - Access control modifier
  - Non Access Modifier

ACCESS CONTROL MODIFIER

❖Java language has four access modifier to control access levels for classes, variable methods and constructor.

- ❖**Default** : Default has scope only inside the same package
- ❖**Public** : Public has scope that is visible everywhere
- ❖**Protected** : Protected has scope within the package and all sub classes
- ❖**Private** : Private has scope only within the classes

ACCESS CONTROL MODIFIER



http://www.4kitsolutions.com/

ACCESS CONTROL MODIFIER

Modifier	Class	Constructor	Method	Data/variables
Public	Yes	Yes	Yes	Yes
Protected		Yes	Yes	Yes
Default	Yes	Yes	Yes	Yes
Private		Yes	Yes	Yes

METHOD OVERLOADING

❖If two or more method in a class have same name but different parameters, it is known as method overloading.

❖Overloading always occur in the same class(unlike method overriding).

❖Method overloading is one of the ways through which java supports polymorphism.

❖Method overloading can be done by **changing number of arguments** or by **changing the data type of arguments**.

What is method signature

❖A method signature is the method name and the number and type of its parameters.

❖Return types and thrown exceptions are not considered to be a part of the method signature.

NON ACCESS MODIFIER

❖Non-access modifiers do not change the accessibility of variables and methods, but they do provide them special properties. Non-access modifiers are of 5 types,

- ❖Final
- ❖Static
- ❖Transient
- ❖Synchronized
- ❖Volatile

FINAL

❖This modifier applicable to **class**, **method**, and **variables**.

❖ This modifier tells the compiler not to change the value of a variable once assigned.

❖ If applied to class, it cannot be sub-classed. If applied to a method, the method cannot be overridden in sub-class.

❖A final method can be inherited/used in the subclass, but it cannot be overriden.

```
final int a = 5;
```

```
class StudyJava{
 final void learn(){System.out.println("learning
something new");
    }
}
```

FINAL

❖A reference variable declared final can never be reassigned to refer to an different object.

❖However, the data within the object can be changed. So, the state of the object can be changed but not the reference.

STATIC

- Static modifier is applicable to:
  - Method
  - Variable
  - Class nested within another class
  - Initialization block

- Static modifier is not applicable to:
  - Class (Not Nested)
  - Constructor
  - Interfaces
  - Method Local Inner Class
  - Inner Class methods
  - Instance Variables
  - Local Variables

STATIC

❖Static Modifiers are used to create **class variable and class methods** which can be accessed **without instance of a class.**

❖Static variables are defined as a class member that can be accessed without any object of that class.

❖Static variable has only one single storage.

❖All the object of the class having static variable will have the same instance of static variable.

❖Static variables are initialized only once.

Static Variable Vs Instance Variable

Static variable	Instance Variable
Represent common property	Represent unique property
Accessed using class name	Accessed using object
get memory only once	get new memory each time a new object is created

STATIC METHOD

- A method can also be declared as static.
- Static methods do not need instance of its class for being accessed.
- main() method is the most common example of static method.
- main() method is declared as static because it is called before any object of the class is created.

STATIC METHOD

Vehicle.java

```
1  package sct;
2
3  public class Vehicle {
4
5      private int doors;
6      private int speed;
7      private String color;
8
9      public static void run(){
10         //Static Run method implementation.
11     }
12
13     public void stop (){
14         //Implementation of Stop method
15     }
16 }
17
18 class Maruti {
19     public void TestVehicleClass(){
20     //To Access run() method we dont need object of Vehicle class
21     Vehicle.run();
22     // To Access stop() method we need object of Vehicle class, else compilation fails.
23     new Vehicle().stop();
24     }
25 }
```

STATIC BLOCK

❖ Java supports a special block, called **static block** (also called static clause) which can be used for static initializations of a class.

❖This code inside static block is executed only once: the first time you make an object of that class or the first time you access a static member of that class (even if you never make an object of that class).

❖Also, static blocks are executed before constructors.

What is the use of static block

- Static block is mostly used for changing the default values of static variables. This block gets executed when the class is loaded in the memory.

- A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program.

NON ACCESS MODIFIERS

❖**Transient modifier**

❖When an instance variable is declared as transient, then its value doesn't persist when an object is serialized

❖**Synchronized modifier**

❖When a method is synchronized it can be accessed by only one thread at a time.

❖**Volatile modifier**

❖The volatile modifier is used to let the JVM know that a thread accessing the variable must always merge its own private copy of the variable with the master copy in the memory.

❖Accessing a volatile variable synchronizes all the cached copied of the variables in the main memory. Volatile can only be applied to instance variables, which are of type object or private.

CONSTRUCTORS

❖A constructor is a special method that is used to initialize an object.

❖It is treated as a special member function because its name is the same as the class name.

❖Java constructors are invoked when their objects are created

❖Every class has a constructor.

❖If we don't explicitly declare a constructor for any java class the compiler builds a default constructor for that class.

❖A constructor does not have any return type.

❖Constructor in Java can not be abstract, static, final or synchronized. These modifiers are not allowed for constructor.

TYPES OF CONSTRUCTORS

❖**Default constructor** (no-arg constructor) - A constructor having no parameter is known as default constructor. It is also known as no-arg constructor.

❖**Parameterized constructor** - A constructor having argument list is known as parameterized constructor.

CONSTRUCTOR OVERLOADING

❖Like methods, a constructor can also be overloaded.

❖Overloaded constructors are differentiated on the basis of their type of parameters or number of parameters.

```
class paramC{
 paramC(int a, int b){
  System.out.print("Parameterized Constructor");
  System.out.println(" having Two parameters");
 }
  paramC(int a, int b, int c){
  System.out.print("Parameterized Constructor");
  System.out.println(" having Three parameters");
 }
}
```

THIS KEYWORD

- this keyword is used to refer to current object.
- this is always a reference to the object on which method was invoked.
- this can be used to invoke current class constructor.
- this can be passed as an argument to another method.

```
class Box
 {
  Double width, height, depth;
  Box (double w, double h, double d)
  {
   this.width = w;
   this.height = h;
   this.depth = d;
  }
}
```

Call Overloaded Constructor

```
public class ThisExample {

	ThisExample() {
			this("Java");
			System.out.println("Inside
Constructor without parameter");
		}
	ThisExample(String str) {
			System.out
				.println("Inside Constructor
with String parameter as " + str);
		}

}
```

- this keyword can only be the first statement in Constructor.
- A constructor can have either this or super keyword but not both.

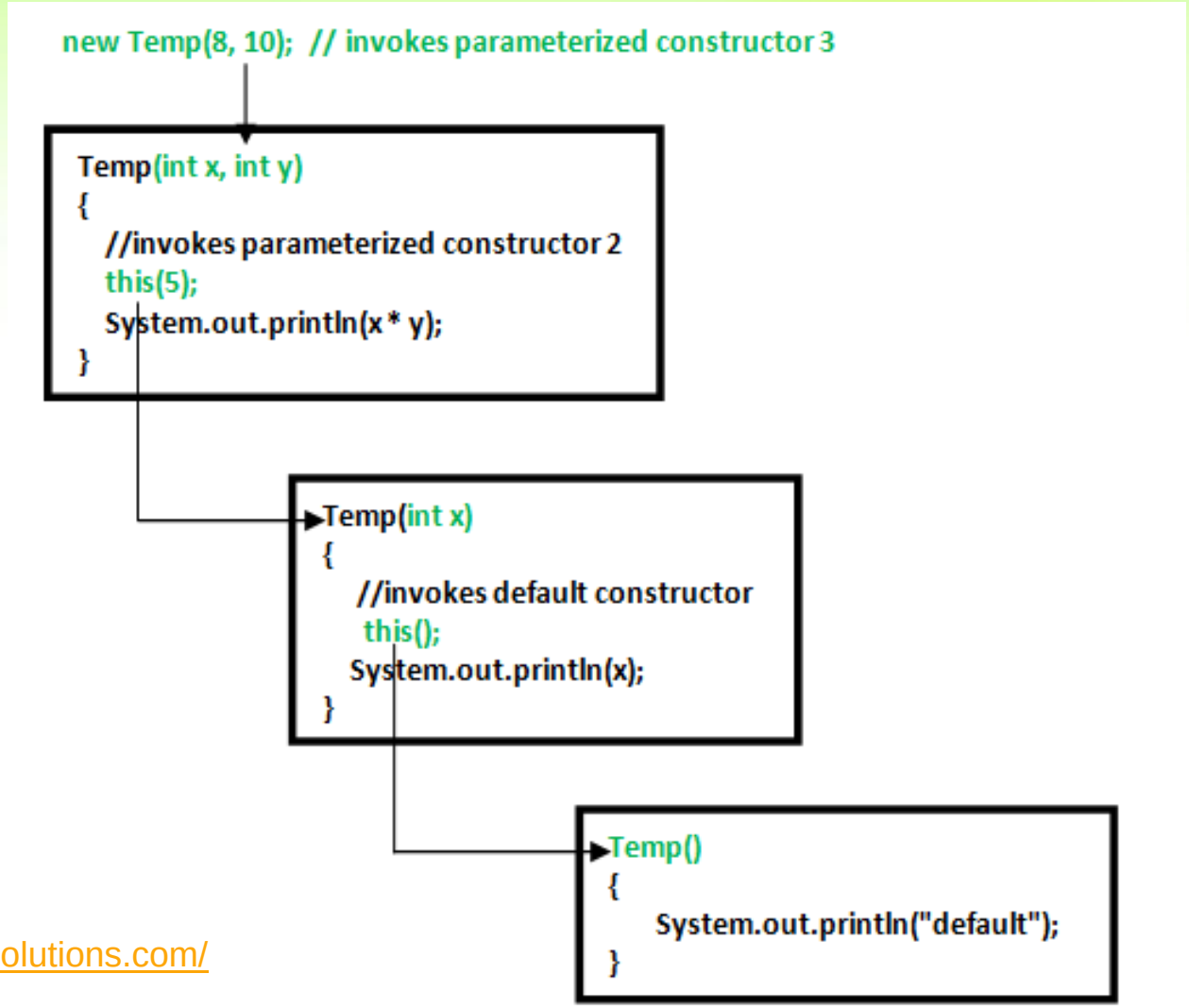
Call method of the class

```
public void getName()
{
 System.out.println("StudyJava");
}

public void display()
{
 this.getName();
 System.out.println();
}
```

THIS AS METHOD PARAMETER

```
void method() {
    method1(this);
  }
  void method1(JBT1 t) {
    System.out.println(t.i);
  }
```

CONSTRUCTOR CHAINING



ARRAYS

❖An array is a collection of similar data types.

❖Array is a container object that hold values of homogenous type.

❖It is also known as static data structure because size of an array must be specified at the time of its declaration.

❖An array can be either primitive or reference type. It gets memory in heap area. Index of array starts from zero to size-1.

❖It occupies a contiguous memory location.

ARRAY DECLARATION

```
datatype[ ] identifier;
or
datatype identifier[ ];
```

```
int[ ] arr;
char[ ] arr;
short[ ] arr;
long[ ] arr;
int[ ][ ] arr;   // two dimensional array.
```

Initialization of Array

❖**new** operator is used to initialize an array.

```
int[ ] arr = new int[10];    //this creates an empty
array named arr of integer type whose size is 10.
or
int[ ] arr = {10,20,30,40,50};  //this creates an
array named arr whose elements are given.
```

ACCESSING ARRAY ELEMENTS

❖Array index starts from 0. To access nth element of an array. Syntax

```
arrayname[n-1];
```

❖To find the length of an array, we can use the following syntax: array_name.length.

FOR EACH LOOP

- J2SE 5 introduces special type of for loop called **foreach** loop to access elements of array.
- Using **foreach** loop you can access complete array sequentially without using index of array.

```
int[] arr = {10, 20, 30, 40};
       for(int x : arr)
       {
```

COPYING ARRAY

❖Copying An Array Using for Loop.

```
int[] a = {12, 21, 0, 5, 7};   //Declaring and
initializing an array of ints

        int[] b = new int[a.length];
//Declaring and instantiating another array of ints
with same length

        for (int i = 0; i < a.length; i++)
        {
            b[i] = a[i];
        }
```

COPYING ARRAY

❖Copying An Array Using **copyOf()** Method of java.util.Array Class

```
int[] a = {12, 21, 0, 5, 7};   //Declaring and
initializing an array of ints

        //creating a copy of array 'a' using copyOf()
method of java.util.Arrays class

        int[] b = Arrays.copyOf(a, a.length);
```

COPYING ARRAY

❖Copying An Array Using **clone()** Method.

❖All arrays will have clone() method inherited from **java.lang.Object** class. Using this method, you can copy an array.

```
int[] a = {12, 21, 0, 5, 7};   //Declaring and
initializing an array of ints

        //creating a copy of array 'a' using clone()
method

        int[] b = a.clone();
```

COPYING ARRAY

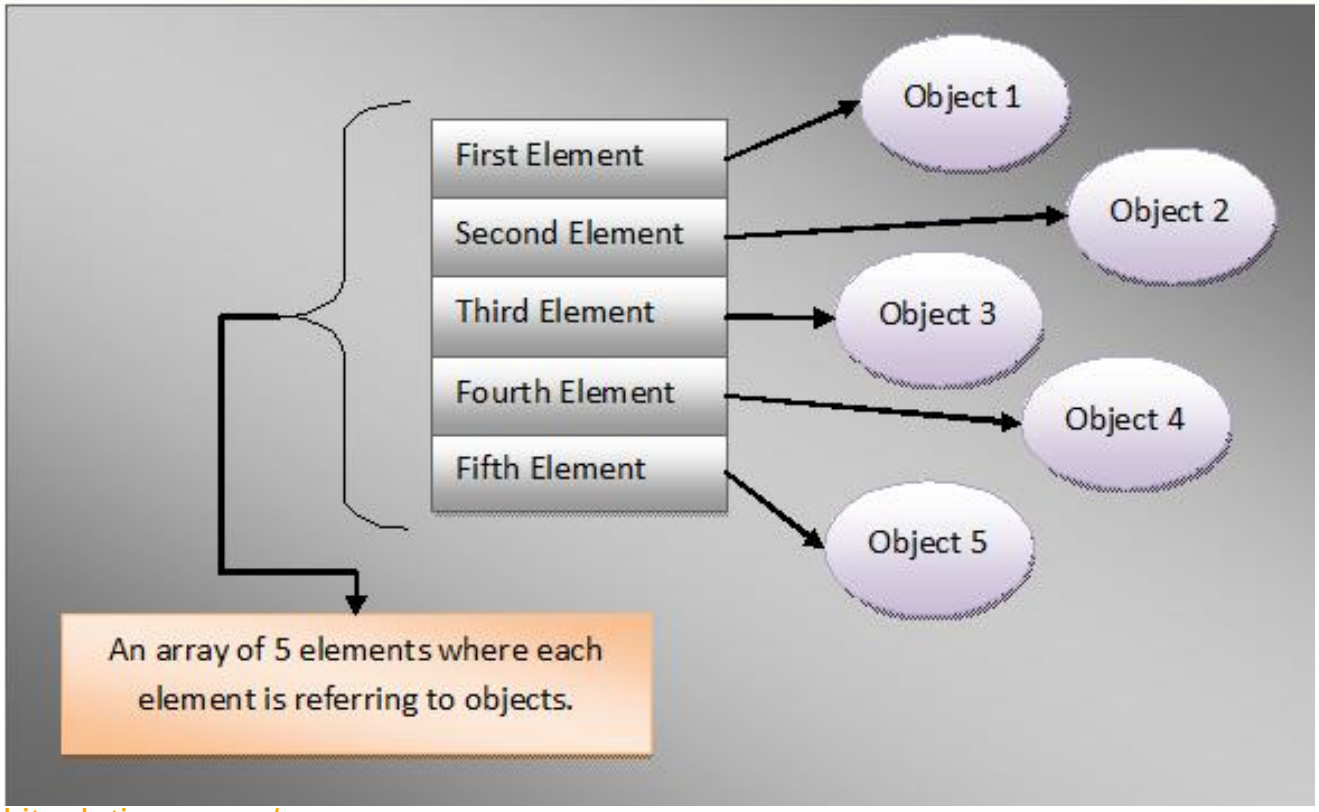
- Copying An Array Using **arraycopy()** Method Of **System Class**.
- Using **arraycopy()** method, you can copy a part of an array into another array.

```
int[] a = {12, 21, 0, 5, 7};   //Declaring and
initializing an array of ints

        //Creating an array object of same length as
array 'a'

        int[] b = new int[a.length];

        //creating a copy of array 'a' using
arraycopy() method of System class

        System.arraycopy(a, 0, b, 0, a.length);
```

ARRAYS OF OBJECTS

❖Array can hold the references to any type of objects.

❖ It is important to note that array can contain only references to the objects, not the objects itself.



Arrays as Parameters

❖Arrays can be passed to method as arguments and methods can return an array.

❖Arrays are **Passed-By-Reference**. That means, When an array is passed to a method, reference of an array object is passed not the copy of the object.

❖So, Any changes made to object in the method will be reflected in the actual object.

MULTI-DIMENSIONAL ARRAY

- A multi-dimensional array is very much similar to a single dimensional array.
- It can have multiple rows and multiple columns unlike single dimensional array

```
datatype[ ][ ] identifier;
int[ ][ ] arr = new int[10][10];
```

JAGGED ARRAY

❖**Jagged arrays** in java are arrays containing arrays of different length.

❖Jagged arrays are also multidimensional arrays. Jagged arrays in java sometimes are also called as **ragged arrays**.

```
int[ ][ ] arr = new int[3][ ];    //there will be 10
arrays whose size is variable
arr[0] = new int[3];
arr[1] = new int[4];
arr[2] = new int[5];
```

INHERITANCE (IS-A)

❖**Inheritance** is one of the key features of Object Oriented Programming.

❖Inheritance provided mechanism that allowed a class to inherit property of another class.

❖When a Class extends another class it inherits all non-private members including fields and methods.

❖Inheritance in Java can be best understood in terms of Parent and Child relationship, also known as **Super class(Parent)** and **Sub class(child)** in Java language.

LAB2

❖Write a program to print the names of students by creating a Student class. If no name is passed while creating an object of Student class, then the name should be "Unknown", otherwise the name should be equal to the String value passed while creating object of Student class.

LAB2

- Create a class named 'Rectangle' with two data members- length and breadth and a method to claculate the area which is 'length*breadth'. The class has three constructors which are :
- 1 - having no parameter - values of both length and breadth are assigned zero.
- 2 - having two numbers as parameters - the two numbers are assigned as length and breadth respectively.
- 3 - having one number as parameter - both length and breadth are assigned that number.
- Now, create objects of the 'Rectangle' class having none, one and two parameters and print their areas.

THANK YOU

http://www.4kitsolutions.com/

By
Sudha Agarwal
sudha.agarwal@4kitsolutions.com