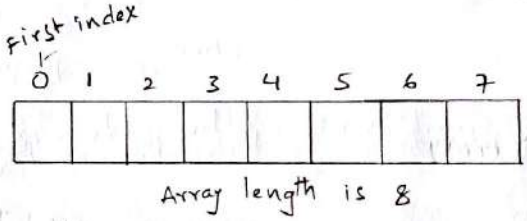
* Arrays:-

→ Till now, we have discussed how to declare variables of a particular datatype, which can store a single value. There are the situations where we might wish to store a group of similar type of values in a variable.

→ It can be achieved by a special kind of data structure known as arrays.

* An array is a collection of similar data elements and it is a data structure where we store similar elements. We can store only fixed set of elements in an array.

→ Array in Java is index based, first element of the array is stored at 0(zero) index.



- Advantages of array:

→ code optimization: It makes the code optimized, we can retrive & sort the data easily.

→ Random access: We can get any data located at any index position.

- Disadvantage of array:

→ Size Limit: We can store only fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java.

There are two types of arrays

- Single dimensional array
- Multi dimensinal array

- **Single dimensional array:**

It is an array with only one dimension (or) index. It can be visualized as a single row or a column.

Declaration:

```
Datatype variable [] = new datatype [size];
(or)
Datatype variable[];
variable = new datatype [size];
```

```
// Declaration & Initialization
(or) datatype variable [] =
     {element1, element2, ... ,n}
```

Ex :- `int marks [] = new int [5];`

(or)

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

(or) `int marks [] = {45, 60, 70, 80, 9};`

Example :-

```
class SingleDarray
{
 public static void main (String args [])
 {
  int marks [] = new int [6]; // Declaration
  marks [0] = 60; // Initialization
  marks [1] = 58;
  marks [2] = 70;
  marks [3] = 80;
  marks [4] = 78;
  marks [5] = 89;
  // printing values in array
  for (int i=0; i < marks.length; i++)
  {
    System.out.println (marks [i]);
  }
}
}
```

(or)

```
int marks [] =
    {60, 58, 70, 80, 78, 89};
```

output:
60
58
70
80
78
89

• Multi dimensional array:

It is an array with two or more dimensions or indexes. It can be visualized as a matrix of rows and columns.

Lets take a two-dimensional array, In this case, data is stored in row and column based index (also known as Matrix form).

Syntax for array declaration (two)

```
Datatype variable [][] = new datatype [rows] [col];
```

(or)

```
Datatype variable[][];
Variable = new datatype [rows] [col];
```

(or)

```
datatype variable [][] =
{{e11, e12 -- } {e21, e22 --} -- }
```

Ex:-

```
int marks[][] = new int [3][6];
```

(or)

```
int marks [][];
marks = new int [3][6];
```

(or)

```
int marks[][] =
{{48, 65 -- } {78, 62 --}{} -- }
```

Example:-

```
class TwoDarray
{
 public static void main(String args[])
{
  int marks [][] = new int[3][6];
  marks [0][0] = 48;
  marks [0][1] = 52;
  marks [0][2] = 63;
        :
        :
  marks [2][4] = 75;
  marks [2][5] = 82;
```

```
// Declaring & Initializing 2D array.
int marks[][] =
{{48,52,63,65,66,71},{67,78,87,65,64,
 76},{78,67,65,77,75,82}};
```

output:-

```
48 52 63 65 66 71
67 78 87 65 64 76
78 67 65 77 75 82
```

```
// printing 2-D array
for (int i=0; i < marks.length; i++) {
for (int j=0; j < marks[i].length; j++) {
      System.out.println (Marks [i] [j] + " ");
    }
    System.out.println ();
} } }
```

* Console input and output (or) Console class :-

→ In this, We learn about java.io.Console class. This class provides convenient methods for reading input and writing output to standard streams (Keyboard and display) in command-line (console) programs. Note:- console class comes under "java.io" package

→ The Console class provides following methods,

those are

- printf() - Writes a formatted string to console's output stream.
- readLine() - Reads a single line of text from console's input stream.
- readPassword() - Reads a password from console input stream with echoing disabled.

Example :-

```
import java.io.*; //package
class ConsoleioDemo
{
public static void main(String args[])
{
Console c= System.console();
c.printf("Enter your name: "); //console output
String name = c.readLine(); //console input
c.printf("Enter your company name : ");
String cname = c.readLine();
c.printf("congrats %s \n", name);
c.printf("you are the Employee of company : %s", cname);
}
}
```

output :-

```
Javac ConsoleioDemo.Java
Java ConsoleioDemo
Enter your name : Madhu
Enter your company name : TKRCET
congrats Madhu
you are the Employee of company : TKRCET
```

* Scanner class :-

→ The Scanner class comes under java.util package. And it is used to getting input from user.

→ System is a class in the java.lang package. This class has three predefined variables : in, out and err.

- in refers to standard input stream (Keyboard)
- out refers to standard output stream (Monitor)
- err refers to standard error output stream (Monitor).

→ The scanner class uses System.in object & variable to get input from Keyboard (Standard input stream).

The scanner class have following Methods

- nextLine() - It returns the input as a string.
- nextInt() - It returns the input as an integer.
- nextFloat() - It returns the input as a Float.
- nextLong() - It returns the input as a long.
- nextShort() - It returns the input as a short.
- nextDouble() - It returns the input as a double.

Example :-

```
import java.util.*;
class ScannerDemo {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter your name: ");
String name = sc.nextLine();
System.out.println("Enter your age : ");
int age = sc.nextInt();
System.out.println("Hai " +name);
System.out.println("your age is " +age);
}
}
```

output :-
```
javac ScannerDemo.java
java ScannerDemo
Enter your name
Madhu
Enter your age
30
Hai, your age is 30
```

* Classes and objects :-

Java is an object-oriented programming language. classes and objects are basic building blocks of OOP.

* classes :-

A class is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind.

(or) A class is a group of objects that has common properties.

→ A class in java can contain: data member, method, constructor, block, class and interface.

Syntax to declare a class

```
class <class_name>
{
 //variables declaration
 // Methods declaration
}
```

→ A class can be declared using the keyword 'class' followed by the name of the class that you want to define.

→ The class body contains two different sections: variable declaration and Methods declaration.

* Objects :-

An object is a software bundle of variables and related Methods. An object is an instance of a class. i.e by using object, we can able access variables and methods in that class.

Syntax to declare an object

```
class_name object_name = new class_name();
```

(or)

In general, Syntax is type objectname;

where type is name of class. because a class is an user-defined data type.

→ The keyword 'new' is used to allocate memory at runtime.

• Instance variable:

A variable that is created inside the class but outside the method, is known as instance variable. Instance variable doesn't get memory at compile time. It gets memory at runtime when object is created. That is why, it is known as instance variable.

Example of object and class:

```
class Student
{
  int id = 521;
  String name = "Madhu";   } //data Members (also instance variables)
  public static void main (String args[]) → //method.
  {
    Student s1 = new Student(); //creating an object of student
    System.out.println ("your R.NO is:" s1.id);
    System.out.println ("your name is:" s1.name);
    int age = 30;
    String branch = "CSE";   } //local variables
    System.out.println ("your age is :" +age);
    system.out.println ("your branch is :" +branch);
  }
}
```

o/p: javac Student.java
java Student
your R.NO is : 521
your name is : Madhu
your age is : 30
your branch is : CSE

In this example, we have created a Student class that have two data members id and name. We are creating the object of the student class by new keyword and printing the objects value.

* Methods:-

Defⁿ: A java method is a collection of statements that are grouped together to perform an operation.

→ None of the methods can be declared outside the class.

→ All methods have a name that starts with a lowercase character.

→ In Java, Methods are used to,

- Make the code reusable.
- Simplify the code.
- Top-down programming.

→ Java supports two types of methods, those are

• Instance methods:- These are used to access/manipulate the instance variables and also access class variables.

• Class methods:- These are used to access class variables but cannot access the instance variables unless and until they use an object for that purpose.

Syntax for Method Declaration

```
[modifiers] return_type method_name (parameter-list)
        {
          statements list // Method body
        }
```

In the above syntax,

* modifiers (optional) defines the scope (public, protected, default or private).
* return_type - It can be either void (if no value is returned) or if a value is returned.
* Method_name - The method name must be a valid Java identifier. (Method name starts with lower case letter).

* Parameter List : you can pass one or more values to a method by listing the values in parantheses following method name.

* Method body : The method body defines what the methods does with the statements.

Example :-

The following example to demonstrate how to define a method and how to call it-

```
import java.io.*;
public class MaxNumber
{
public static void main (String args[])
{
  Console c = System.console();
  c.printf("Enter first number \n");
  int a = Integer.parseInt(c.readLine());
  c.printf("Enter second number \n");
  int b = Integer.parseInt(c.readLine());
  int maxc = MaxFunction(a,b);
  System.out.println("Maximum Number is = " + maxc);
 }
public static int maxFunction(int n1, int n2)
{
   int max;
   if (n1 > n2)
      max = n1;
   else
      max = n2;
   return max;
 }
}
```

O/p: Javac MaxNumber.Java
java MaxNumber
Enter first Number
56
Enter second Number
51
Maximum Number is = 56.

Example: A method with void return type.

```
import java.io.*;
Public class Grade
{
 public static void main (String args[])
 {
  Console  C = System.consde();
  C.printf("Enter your marks in blw 0 to 100 \n");
  int marks = Integer.parseInt(c.readLine());
  grade(marks);
 }
 public static void grade (int tmarks)
 {
   if (tmarks  >= 90)
     System.out.println (" Grade A");
   else if (tmarks >= 70)
     System.out.println(" Grade B");
   else if (tmarks >= 50)
     System.out.println (" Grade C");
   else
     System.out.println(" Grade D");
  }
}
```

o/p: javac Grade.java
java Grade
Enter your marks in blw 0 to 100
92
Grade A.

Note:- If a method is declared using static keyword, then no need to create an object to access it. otherwise, we can create object and then access methods by using that object.

Example: The following example shows method calling with an object creation.

```
import java.io.*;
class Calc
{
public static void main(String args[])
{
 Calc obj = new Calc(); // object creation for class Calc.
 Console c = System.console();
 c.printf("Enter first number");
 int x = Integer.parseInt(c.readLine());
 c.printf("Enter second number");
 int y = Integer.parseInt(c.readLine());
 obj.sum(x,y);
 obj.sub(x,y);
}
void sum(int a, int b)
 {
  System.out.println("sum is :" +(a+b));
 }
void sub(int a, int b)
 {
  System.out.println("sub is :" +(a-b));
 }
}
```

O/P:- javac Calc.java

```
java Calc
Enter first number
20
Enter second number
5
Sum is : 25
sub is : 15
```