### Methods in java

A method is a self contained block of statements to perform some specific task . (It can be comparable with the function in c)

A method needs to be called by other method whenevere required. It is a sub program.

Advantages

Reusability: once it is written,it can be called as many times as it required. maintainability :

easy to debug Types of method

In-built method : Exist in java library.It needs to be called. All methods are associated with the pre defined class.

System.out.println();

nextInt() method of Scanner class User defined method : defined by the programmer

**User defined method**

**Declaration of method**

Every method is identified by its signature or declaration or prototype, ie every method has a unique mehod signature. This is used by either compiler or jvm for binding.

**access specifier modifier return type method name(parameter list)**

**{**

**body of the method**

**}**

**question**

**display reverse of a numbe int reverse(int num)**

**{**

**}**

**copy one array to another array int[] copyarray(int[] source)**

**{**

**}**

**prime check**

**boolean isprime(int num)**

**{}**

**search an element from array**

**{45 ,76,87,34,8}**

**int search(int[] n, int k)**

**{}**

**Access Specifier/visibility mode**: It specifies the visibility of the method(. Java provides four types of access specifier: **public,private,protected,default/package private**

**Modifier:**It is the access type of the method. **static or non-static(Instance method)**

**Return Type:** It is a data type that the method returns(**It is the data type of the result).** It may have a primitive data type, object, collection, void, etc. If the method does not return anything, we use void keyword**.**

**Every method always returns a single value.**

**If the method has return type , then there must be a return statement in the method. return result;**

**Method Name:** It is a unique name or Identifier that is used to define the name of a method.

**Parameter List:** It is the list of parameters separated by a comma and enclosed in the pair of parentheses. These are the input to the method,

It contains the data type and variable name.

If the method has no parameter, left the parentheses blank.

While giving parameter in the method the no of parameter,datatype of parameters,order of parameters matters.

void search(int[] n,int element){

}

int nump[]={5,8,6,9}; search(nump,6);

**User defined method**

**The method written by the user is known as user defined method. Every method has 3 components**

**Method declaration: It is the method signature method definition: Writing the logic of the method**

**method calling: To call the method by other method for specific purpose**

**Rules for calling a method**

**Before calling a method two things are considered ,**

1. **Whether the method is static or non static**
2. **Whether the method has return type or void If it is a static method**

**A method that has static keyword is known as static method. In other words, a method that belongs to a class rather than an instance of a class is known as a static method. We can also create a static method by using the keyword static before the method name.**

**static method can be called without creating an object. It can access static data members and also change the value of it. It is invoked by using the class name.**

**static double mean(int a,int b,int c) //method signature**

**{**

**double res=(a+b+c)/3; //mehod body**

**return res; // if return type there ie other than void then return statement must be there**

**}**

**If it is instance method**

**The method of the class is known as an instance method if it does not have static keywor. It is a non- static method defined in the class. Before calling or invoking the instance method, it is necessary to create an object of its class to call non static method.**

**double mean(int a,int b,int c)**

**{**

**double res=(a+b+c)/3; return res;**

**}**

**If the method has return type the calling of the method is always an expression datatype variable=methodname(arguments);**

**double res=objectname.mean(12,67,54); // non static and return type is there**

**or**

**System.out.println(objectname.mean(12,67,54)); The datatype of variable is same as that of return type.**

**If the method has return type void then the calling of the method is a simple statement. methodname(arguments);**

**objectname.mean(12,78,43); // non static and return type is void(no return type)**

**Note: The methods can be called either by class name or object name , that depends upon whether the method is a static method or non static method.**

**Example for static method or class method class MethodDemo**

**{**

**static double mean(int a,int b,int c)**

**{**

**double res=(a+b+c)/3; return res;**

**}**

**public static void main(String[] s)**

**{**

**double m=mean(23,87,56); // one static method calls another static method directly in same class System.out.println("mean is:"+m);**

**}**

**}**

**Example for non static or instance method class MethodDemo1**

**{**

**double mean(int a,int b,int c)**

**{**

**double res=(a+b+c)/3; return res;**

**}**

**public static void main(String[] s)**

**{**

**MethodDemo1 ob=new MethodDemo1(); //create object first double m=ob.mean(23,87,56);**

**System.out.println("mean is:"+m);**

**}**

**}**

**If return type void class MethodDemo2**

**{**

**void mean(int a,int b,int c)**

**{**

**double res=(a+b+c)/3; System.out.println("mean is:"+res);**

**}**

**public static void main(String[] s)**

**{**

**MethodDemo2 ob=new MethodDemo2(); ob.mean(23,87,56);**

**}**

**}**

**Questions(by using method)**

**write a program to display the prime factors of a number Display the gcd of two numbers**

**Input a number and check whether it is prime or not**

**Input a character and display it in capital letter if it is in small and vise versa display the fibonacci series upto a specific term**

**instanceof**

**objectname instanceof class/interface :returns boolean value ie either true or false**

**compound operator**

**+=.-=.\*=./=.%= a+=1;**

**Ternary operator/conditional operator**

**?:**

**String s=(n%3==0 &&n%5==0)?"divisible":"not divisible";**

**Method Overloading**

**In a class ,we can have more than one method having same name but different signature used for different purpose is known as method overloading.**

**It is polymorphism in oops.**

**Example**

**int addition(int x,int y)**

**{**

**return x+y;**

**}**

**int addition(int a,int b,int c)**

**{**

**return a+b+c;**

**}**

**int addition(int[] n)**

**{**

**int sum=0;**

**for(int i=0;i<n.length;i++) sum+=n[i];**

**return sum;**

**}**

**String addition(String s1,String s2)**

**{**

**return s1+" "+s2;**

**}**

### How to pass an array to a method

**Every method returns a single value Method declaration**

**int[] num; //array declaration num=new int[8]; //definition of arry**

void display(int[] arr) //method declaration

{

for(int i=0;i<arr.length;i++) System.out.println(arr[i]);

}

**call**

**while calling a method , the reference name of the array is passed**

display(n);

void quicksort(int[] num,int l,int u)

{

}

void heapify(int[] n,int size)

{

}

**Returning array from a method**

int[] copy(int[] s)

{int[] n=new int[s.length]; for(int i=0;i<s.length;i++) n[i]=s[i];

return n; //reference is returned

}

**Exa**

class copy

{

static void display(int[] arr)

{

for(int i=0;i<arr.length;i++) System.out.println(arr[i]);

}

// how to return an array from a method

static int[] copy(int[] s)

{

int[] n=new int[s.length]; for(int i=0;i<s.length;i++) n[i]=s[i];

return n;

}

public static void main(String[] s)

{

int[] num1={10,20,30,40,40};

int[] n=new int[num1.length];

n=copy(num1);

display(n);

//copy using predefined method arracopy System.arraycopy(num1,0,n,0,4);

display(n);

// copy using clone() method int[] n2=n.clone(); display(n2);

}

}

**pass a 2-D array to a method**

class Arraydemo1

{

static void display(int[][] a)

{

for(int i=0;i<a.length;i++) //mat.length is the row size

{

for(int j=0;j<a[1].length;j++) //mat[0].length is the column size

{ System.out.print(a[i][j]+" ");

}

System.out.println("¥n");

}

}

public static void main(String[] s)

{

int[][] mat=new int[2][2]; mat[0][0]=12;

mat[0][1]=16;

mat[1][0]=17;

mat[1][1]=10;

display(mat);

}

}

**Recursion method**

**When a method calls itself upto meet the base condition.**

**Base condition is the condition ,when it satisfies, the method calling terminates. the updated parameters are passed in the method**

**int factorial(int n)**

**{**

**if(n==0||n==1) return 1;**

**else return(n\*fact(n-1));**

**}**

**int[] duplicate(int[] n)**

**{**

**} [45,76,2,78,45,2,76]**

**[45,76,2]**

**class mataddition**

**{**

**int[][] add\_matrix(int[][]m1,int[][]m2)**

**{**

**int[][]m3=new int[100][100]; for(int i=0;i<m1.length;i++) for(int j=0;j<m1[0].length;j++)**

**m3[i][j]=m1[i][j]+m2[i][j]; return m3;**

**}**

**void display(int[][] a)**

**{**

**for(int i=0;i<a.length;i++) //mat.length is the row size**

**{**

**for(int j=0;j<a[1].length;j++) //mat[0].length is the column size**

**{ System.out.print(a[i][j]+" ");**

**}**

**System.out.println("¥n");**

**}**

**}**

**public static void main(String[] s)**

**{**

**mataddition o=new mataddition(); int[][]m1={{2,4,9},{6,7,9}};**

**int[][]m2={{12,14,9},{16,17,9}};**

**int[][]res=new int[10][10]; res=o.add\_matrix(m1,m2); o.display(res);**

**}**

**}**

### How to pass an array to a method

**Every method returns a single value Method declaration**

**int[] num; //array declaration num=new int[8]; //definition of arry**

void display(int[] arr) //method declaration

{

for(int i=0;i<arr.length;i++) System.out.println(arr[i]);

}

**call**

**while calling a method , the reference name of the array is passed**

display(n);

void quicksort(int[] num,int l,int u)

{

}

void heapify(int[] n,int size)

{

}

**Returning array from a method**

int[] copy(int[] s)

{int[] n=new int[s.length]; for(int i=0;i<s.length;i++) n[i]=s[i];

return n; //reference is returned

}

**Exa**

class copy

{

static void display(int[] arr)

{

for(int i=0;i<arr.length;i++) System.out.println(arr[i]);

}

// how to return an array from a method

static int[] copy(int[] s)

{

int[] n=new int[s.length]; for(int i=0;i<s.length;i++) n[i]=s[i];

return n;

}

public static void main(String[] s)

{

int[] num1={10,20,30,40,40};

int[] n=new int[num1.length];

n=copy(num1);

display(n);

//copy using predefined method arracopy System.arraycopy(num1,0,n,0,4);

display(n);

// copy using clone() method int[] n2=n.clone(); display(n2);

}

}

**pass a 2-D array to a method**

class Arraydemo1

{

static void display(int[][] a)

{

for(int i=0;i<a.length;i++) //mat.length is the row size

{

for(int j=0;j<a[1].length;j++) //mat[0].length is the column size

{ System.out.print(a[i][j]+" ");

}

System.out.println("¥n");

}

}

public static void main(String[] s)

{

int[][] mat=new int[2][2]; mat[0][0]=12;

mat[0][1]=16;

mat[1][0]=17;

mat[1][1]=10;

display(mat);

}

}

**Recursion method**

**When a method calls itself upto meet the base condition.**

**Base condition is the condition ,when it satisfies, the method calling terminates. the updated parameters are passed in the method**

**int factorial(int n)**

**{**

**if(n==0||n==1) return 1;**

**else return(n\*fact(n-1));**

**}**

**int[] duplicate(int[] n)**

**{**

**} [45,76,2,78,45,2,76]**

**[45,76,2]**

**class mataddition**

**{**

**int[][] add\_matrix(int[][]m1,int[][]m2)**

**{**

**int[][]m3=new int[100][100]; for(int i=0;i<m1.length;i++) for(int j=0;j<m1[0].length;j++)**

**m3[i][j]=m1[i][j]+m2[i][j]; return m3;**

**}**

**void display(int[][] a)**

**{**

**for(int i=0;i<a.length;i++) //mat.length is the row size**

**{**

**for(int j=0;j<a[1].length;j++) //mat[0].length is the column size**

**{ System.out.print(a[i][j]+" ");**

**}**

**System.out.println("¥n");**

**}**

**}**

**public static void main(String[] s)**

**{**

**mataddition o=new mataddition(); int[][]m1={{2,4,9},{6,7,9}};**

**int[][]m2={{12,14,9},{16,17,9}};**

**int[][]res=new int[10][10]; res=o.add\_matrix(m1,m2); o.display(res);**

**}**

**}**