Functions/Methods

Functions are class members used to perform task/operation. Functions are used to achieve modularity and reusability.

Syntax
```
<Access    <Access     <return   <Function    <Function
specifier>  modifier>    type>      name>      paramtere>
{
----
-----
}
```

Function parameters

Function parameters are local variables declared in function declaration line. Function parameters are used to hold function arguments. Function can be declared with multiple function parameters and it should be separated by a comma.

Function arguments

The value which are assigned to function parameters are called as function arguments.

Function return type

We can return value from the function using return statements. Return statements should be declared using keyword return. If we are returning value from the function, we have to specify the data type of the value in the return type field.
We cannot return multiple values from the function. if function is not retuning any value then return type should be mentioned as void.

```
Class Demo
{
Public static void add( )
{
int x=20+30;
System.out.println("add value is" +x);
}

Public static void main(String args[])
{
System.out.println("Main method start");
add();  // function calling
System.out.println("Main method end");
}
}
}
```

1. Write a prog to swap two numbers passing parameter
```
class Function5
{

void swap(int a,int b)
{
```

```
int temp;
System.out.println (Value of a before swap"+a);
System.out.println (Value of b before swap"+b);

temp=b;
b=a;
a=temp;

System.out.println (Value of a after swap"+a);
System.out.println (Value of b after swap"+b);
}

public static void main(String args[])
{
swap(10,20);
}
}
```

2. Write a prog to swap two numbers without using temp

```
class Function6
{
static void swap(int a,int b)
{
System.out.println ("Value of a before swap"+a);
System.out.println ("Value of b before swap"+b);

a=a+b;
b=a-b;
a=a-b;

System.out.println ("Value of a after swap"+a);
System.out.println ("Value of b after swap"+b);
}
public static void main(String args[])
{
swap(10,20);
}
}
```

Passing value to the function

1. Static void add(int a,int b)

```
{
int c=a+b;
}
Psvm()
{
add(10,20);
}
```

2. Void m1(int x)

```
{
}
Psvm()
{
m1(2.06) \\error (should pass integer value)
}
```

3. Void m1(int x,int y)
   {
   }
   PSVM( )
   {
   m1(10); \\error only one argument is passed
   }

4. Void m2(char ch)
   {
   }
   Ch(); \\error some argument should be passed

3. Write a prog to perform login operation

```
class Function1
{
static void Login(String un, String pwd)
{
if((un=="Alex") && (pwd=="1234"))
{
System.out.println("Login successfull");
}
else
{
System.out.println("Login unsuccessfull");
}
}

public static void main(String[] args)
{
Login("Alex","1234");
}

}
```

4. Write a prog to display value between parameters

```
class Function7
{
static void print(int a, int b)
{
for(int i=a;i<=b;i++)
{
System.out.println(i);
}
System.out.println("The value between "+a+ " and "+ b);
}
public static void main(String args[])
{
print(1,10);
}
}
```

5. Write a prog to print square of all numbers between 1 to 10 using function argument

```
class Function3
{
public static void square(int n)
{
System.out.println("Square of "+n+ "is"+n*n);
}
public static void main(String args[])
{
for (int i=1;i<=10;i++)
{
square(i);
}
}
}
```

6. Write a prog to print factorial of all number btw 1 to 4 using function parameter

```
class Function2
{
public static void factorial(int n)
{
int fact=1;
for ( int x=1;x<=n;x++)
{
fact=fact*x;
}
System.out.println("Factorial of "+n+ "is"+fact);
}
public static void main(String args[])
{
for (int i=1;i<=4;i++)
{
factorial(i);
}
}
}
```

Function recursion

The process of calling function within itself is known as function recursion.

```
Class Demo
{
Static void m1()
{
SYstm.out.println("The method m1");
m1( );  \\stack overflow
}
Public static void main(String args[])
{
m1();
}
}
```

When function is called within itself without any condition then stack overflow will occur, to avoid this add a condition for that function.

Eg:

```
class Function11
{
static int x=1;
static void m1()
{
if(x<=2)
{
System.out.println("The method is m1");
x++;
m1();
}
}

public static void main(String args[])
{
m1();
}
}
```

1. Write a prog to print numberfrom 1 to 10 without using loop

```
class Function12
{

static int x=1;
static void print()
{
if(x<=10)
{
System.out.println(x);
x++;
print();
}
}

public static void main(String args[])
{
print();
}
}
```

2. Write a prog to display all even umbers between 20 to 30 without using loop

```
class Function13
{
static int x=20;
static void print()
{
if(x<=30)
{
```

```
if(x%2==0)
{
System.out.println("The value of"+x+" is even" );
}
x++;
print();
}
}

public static void main(String args[])
{
print();
}
}
```

3. Write a prog to display sum of even n odd numbers without using loop

```
class Function8
{

static int x=10;

static int evensum=0,oddsum=0;

static void m1()
{

if(x<=20)

{

if(x%2==0)
{
evensum=evensum+x;
}

else
{
oddsum=oddsum+x;
}

x++;

m1();

}

}

public static void main(String args[])
{
m1();
System.out.println("Even sum is" +evensum);
System.out.println("odd sum is "+oddsum);

}
}
```

Returning value from function

1. void m1( )
   ```
   {
   Return 10; \\error
   }
   ```
   Void can return only value as 0;

2. void m1()
   ```
   {
   return 0;
   }
   ```

3. int m1( )
   ```
   {
   }\\error
   ```
   Since return type is specified return should be included in the statement.

4. int m1()
   ```
   {
   Return 1.2; \\error
   }
   ```
   Since data type of return type is not matching with value returning

5. int m1()
   ```
   {
   Return 10,20,30; \\error
   }
   ```
   Only one value can be returned.

6. int m1()
   ```
   {
   Return 10; \\error
   Return 20;
   Return 30;
   }
   ```
   Only one return statement.

7. int m1()
   ```
   {
   Int x=1;
   If(x<=1)
   {
   Return 1;
   }
   Else
   {
   Return 0;
   }
   }
   ```
   Since at a time only one return statement is executed this is correct.

```
//Declaring a return type


class Function14
{

static int add( int x, int y)
{

int z=x+y;
return z;
}

public static void main(String args[])
{

int a;

a=add(10,20);

System.out.println("The sum of a and b is"+a);
}
}
```

2. Prog using boolean as return type

```
class Function15
{
static boolean vote( String name, int age)
{
if(age>=18)
{
return false;
}
else
{
return true;
}
}

public static void main(String args[])
{
String name="Alex";
int age=16;
 boolean res=vote(name,age);
if(res==false)
{
System.out.println(" Person can vote");
}
else
{
System.out.println(" Person cant vote");
}
}

}
```