Return type:

(i) When function will complete its execution then the calling function. ↑ call is have to going some value.

(ii) The value is going to be whatever you return in the function.

(iii) If no return type then write void.

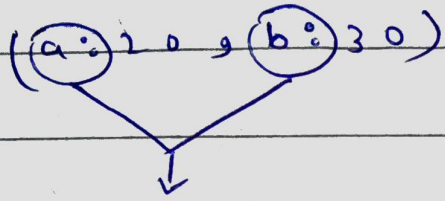
int + int = int

remember explicity.

(iv) Ater execution of return statement the function will terminate its execution.

Day: _____ Date: _____

(*) Pass the value of variable when you are calling the method in main()

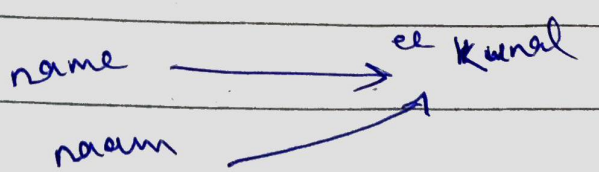


for highlighting, feature of intellije ideer.

Swap code :

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

★ There is no pass by reference only pass by value.



★ copy of name.

★ pointing toward another

★ creating a new object we are

Day: Date:

★ primitives :

int, short, char, byte ---

↓

just passing value.

★ object & stuff : passing value of refrence.

~~a → 10~~

swap(a,b)

a → 10

num1 → 10

swap(num1,num2)

b → 20

num2 → 20

only value will passed.

a → 10

num1 → 10

→ save as a → 10

num1 ↗

★ array are pass ~~by~~ refrence not pass by value.

Day: Date:

Shadow:

Shadow begins when the local variable is declared.

scope will begin when the value is initialized.

Firstly programme will access the lower level variable.

```
Public class Name
{
static int x = 20;
public static void main(String[] args)
{
    int x;
    x = 20
    System.out.print(x);
}
}
```

This will access lower level. It will remove this (x=20)

compiler will access lower level and this error

Day: Date:

VarArgs:

When we dont know how many variable will pass then VarArgs take place. or we can say this var args

int ...v → variable

int → data type

... → dot

v → variable

internally it is taking integers in array.

length is not constant. This may be vary.

int ...v

↓

array of type integer.

they should always come at end.

at last.

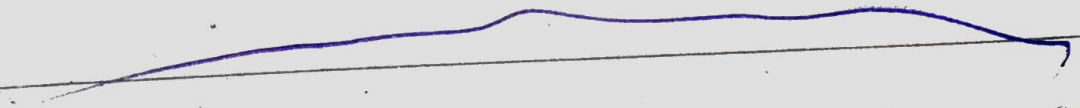
varArg(int a, int b, int ...v)

{

ay: _____ Date: _____

Function overloading:

* Number of arguments and types of argument should be different for the functions that have the same name.



vartr cannot be empty. in overloading.