METHODS

SYNTAX

data type of the result

Any Valid Identifier

RETURN-VALUE-TYPE METHOD-NAME(PARAMETER-LIST)
{
DECLARATIONS AND STATEM
}

comma separated list, declares parameters.

void - method returns nothing

Method Can return at most one value

TYPES OF METHOD

Need Object

Can Run Without Object

- ❑ INSTANCE METHOD
- ❑ STATIC/CLASS METHOD

TYPES OF METHOD

```
PUBLIC CLASS METHOD1 {
    VOID DISPLAY1()     {
        SYSTEM.OUT.PRINTLN("INSTANCE METHOD");
    }
    STATIC VOID DISPLAY2()      {
        SYSTEM.OUT.PRINTLN("STATIC OR CLASS METHOD");
    }
    PUBLIC STATIC VOID MAIN(STRING ARGS[])      {
        METHOD1 OBJ=NEW METHOD1();
        OBJ.DISPLAY2();        // STATIC OR CLASS METHOD CALLING
        METHOD1.DISPLAY2();    // STATIC OR CLASS METHOD CALLING
        DISPLAY2();            // STATIC OR CLASS METHOD CALLING
        OBJ.DISPLAY1();        // INSTANCE METHOD CALLING        } }
```

METHOD WITH ARGUMENTS

```
PUBLIC CLASS              {
    INT K;
    VOID COUNT(INT N)
    {
     WHILE(N>0)       {
         K++;
         N=N/10;
                         }
SYSTEM.OUT.PRINTLN("NUMBER OF DIGITS : "+K);
     }
    PUBLIC STATIC VOID MAIN(STRING ARGS[])     {
        METHOD2 OBJ=NEW METHOD2();
        OBJ.COUNT(823);        }   }
```

823

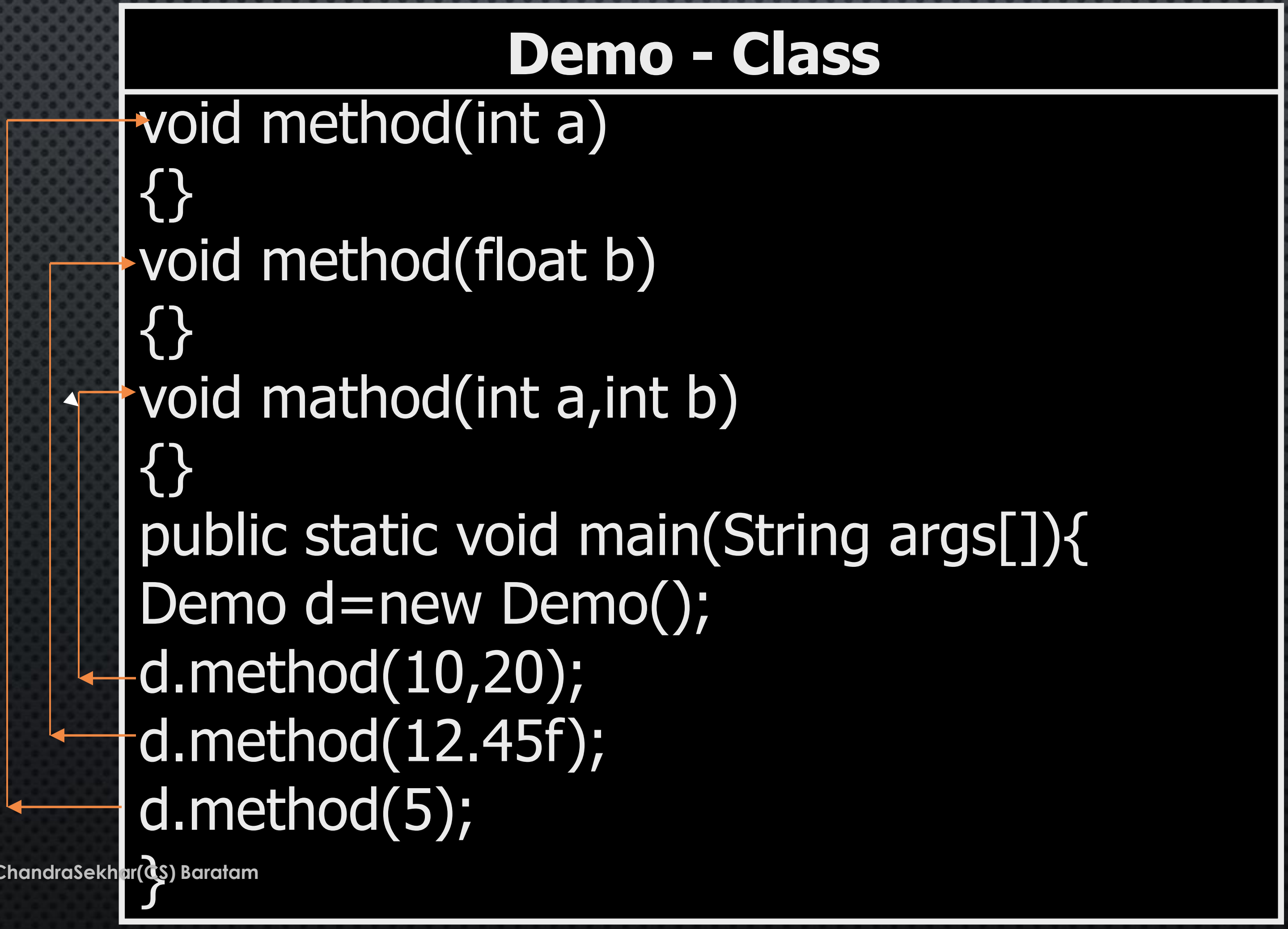
```
823>0  True
k++ => k=1
n=823/10=> n=82
82>0  True
k++ => k=2
n=82/10 => n=8
8>0 True
k++ => k=3
n=8/10 => n=0
n>0   False
```

ChandraSekhar(CS) Baratam

METHOD WITH RETURN

```
PUBLIC CLASS METHOD3 {
   INT K;
   INT COUNT(INT N)
   {
     WHILE(N>0)       {
         K++;
         N=N/10;          }
     RETURN K;
   }
   PUBLIC STATIC VOID MAIN(STRING ARGS[])     {
       INT N;
       METHOD3 OBJ=NEW METHOD3();
       N=OBJ.COUNT(823);
       SYSTEM.OUT.PRINTLN("NUMBER OF DIGITS : "+N);       }   }
```

METHOD OVERLOADING



Demo - Class

```
void method(int a)
{}
void method(float b)
{}
void mathod(int a,int b)
{}
public static void main(String args[]){
Demo d=new Demo();
d.method(10,20);
d.method(12.45f);
d.method(5);
}
```

METHOD OVERLOADING

```
PUBLIC CLASS NEWCLASS  {
VOID FUN(INT N)     {
    INT F=1;
    FOR(INT I=1;I<=N;I++)              F=F*I;
    SYSTEM.OUT.PRINTLN("FACTORIAL OF "+N+" IS :
     "+F);       }
VOID FUN(INT A,INT B)      {
   SYSTEM.OUT.PRINTLN("BEFORE SWAPPING");
   SYSTEM.OUT.PRINTLN("A : "+A+" B : "+B);
   A=A+B;          B=A-B;          A=A-B;
   SYSTEM.OUT.PRINTLN("AFTER SWAPPING");
   SYSTEM.OUT.PRINTLN("A : "+A+" B : "+B);       }
VOID FUN(FLOAT B)      {
   SYSTEM.OUT.PRINTLN("SQUARE : "+(2*B));       }
```

```
PUBLIC STATIC VOID
     MAIN(STRING ARGS[])      {
NEWCLASS OBJ=NEW
     NEWCLASS();
      OBJ.FUN(3.5F);
     OBJ.FUN(5);
     OBJ.FUN(10,20);
    }
   }
```

METHOD OVERLOADING

```
public class NewClass  {
void fun(int n)     {
    int f=1;
    for(int i=1;i<=n;i++)           f=f*i;
    System.out.println("Factorial of "+n+" is : "+f);       }
void fun(int a,int b)     {
   System.out.println("Before Swapping");
   System.out.println("A : "+a+" B : "+b);
   a=a+b;          b=a-b;          a=a-b;
   System.out.println("After Swapping");
   System.out.println("A : "+a+" B : "+b);       }
void fun(float b)     {
   System.out.println("Square : "+(2*b));       }
```

```
public static void main(String args[])      {
NewClass obj=new NewClass();
     obj.fun(3.5f);
    obj.fun(5);
    obj.fun(10,20);
   }
  }
```

METHOD OVERLOADING

```
public class NewClass  {
void fun(int n)     {
    int f=1;
    for(int i=1;i<=n;i++)             f=f*i;
    System.out.println("Factorial of "+n+" is : "+f);        }
void fun(int a,int b)     {
   System.out.println("Before Swapping");
   System.out.println("A : "+a+" B : "+b);
   a=a+b;          b=a-b;          a=a-b;
   System.out.println("After Swapping");
   System.out.println("A : "+a+" B : "+b);       }
void fun(float b)     {
   System.out.println("Square : "+(2*b));       }
```

```
public static void main(String args[])       {
NewClass obj=new NewClass();
    obj.fun(3.5f);
    obj.fun(5);
    obj.fun(10,20);
  }
 }
```

METHOD OVERLOADING

```
public class NewClass  {
void fun(int n)     {
    int f=1;
    for(int i=1;i<=n;i++)              f=f*i;
    System.out.println("Factorial of "+n+" is :
     "+f);       }
void fun(int a,int b)      {
   System.out.println("Before Swapping");
   System.out.println("A : "+a+" B : "+b);
  a=a+b;          b=a-b;          a=a-b;
   System.out.println("After Swapping");
   System.out.println("A : "+a+" B : "+b);      }
void fun(float b)      {
   System.out.println("Square : "+(2*b));      }
```

```
public static void
     main(String args[])      {
NewClass obj=new
     NewClass();
       obj.fun(3.5f);
     obj.fun(5);
     obj.fun(10,20);
    }
   }
```

Access Modifiers

VISIBILITY MODIFIERS

- <u>Public:</u>

  if the method or variable must be visible to all classes, then it must declared as public.

- <u>Private:</u>

  It is narrowly visible and highest level of protection. Private methods and variables cannot seen by any class other than the one in which they are defined.

- <u>Protected:</u>

  This modifier is a relationship between a class and its present and future subclasses.

- <u>Package:</u>

  package is indicated by the lack of any access modifier in a declaration. It has an increased protection and narrowed visibility

Scope

Specifier	class	subclass	package	world
private	✓			
protected	✓	✓	✓	
public	✓	✓	✓	✓
package	✓		✓	

Class & Objects

CLASS

A *CLASS* IS A COLLECTION OF *FIELDS* (DATA) AND *METHODS* (PROCEDURE OR FUNCTION) THAT OPERATE ON THAT DATA.

SYNTAX:

```
CLASS CLASS-NAME
    {

    }
```

KeyWord

Valid Identifier

Creating Objects of a Class

Objects are created dynamically using the *new* keyword.

Syntax:

Class-name object-name=new Class-name();

ACCESSING DATA

```
ObjectName.VariableName

ObjectName.MethodName(parameter-list)
```