

Object Oriented Programming JAVA

Dr. Prafulla Kalapatapu
Computer Science Engineering
Mahindra Ecole Centrale
prafulla.kalapatapu@mechyd.ac.in



**Mahindra
École Centrale**
COLLEGE OF ENGINEERING

ENCAPSULATION

Encapsulation

- **What is an Encapsulation?**
Combining data and code at one place as a module.
- **How can we manage the data?**
Using variables/fields/attributes
- **How can we manage the code?**
Using methods/ functions
- **How can we implement Encapsulation in Java?**
Using classes and objects

- **What is a class?**
 - Class is an user defined type, which contains variables and methods.
 - It is a blue print/template for creating objects.
 - It doesnt exist physically (i.e memory wont be allocated).

- **What is an Object?**

Object is an instance of class

- **Syntax**

Keyword ← class classname → User defined name

Class body {
Variables/fields/properties
+
Methods/operations/behaviour
}

Name Conventions

- **For Class Name :**
 - Class name starts with a capital letter **Ex: Employee, Student**
 - If class name contains multiple words each word should start with capital letter.
Ex: HelloWorld
- **For Variables :**
 - Variable names should be in small letters. **Ex: int total;**
- **For Constants:**
 - Constant variable names should be in capital letters
Ex: final int PI=3.14;
- **For Methods:**
 - Method name starts with a small letter
 - From second word onwards, each word's first letter will be in capital letter
Ex: taxCalculation();

Variables

- Purpose is to store the data
- Variables
 - 1. Instance variables/object level/attributes
 - 2. Static variables/class level.

1. Instance Variables:

If we declare any variable in class, then we can call those variables as instance variables.

- When memory allocates for instance variable?

When we create an instance/object, at that time, memory will allocate for instance variable.

- Can we use multiple instances for a class?

Yes (any no.of)

Variables [1]

2. Static Variables:

If we declare any variable with a keyword “static”, in a class, we can call those variables as static variables.

- When memory allocates for static variable?

When the class is loaded into the memory, at that time, memory will be allocated to static variable.

- Similarities between instance and static variables

Both are used to store data

Variables [2]

- Difference between instance and static variables.

Instance Variables	Static Variables
No static keyword	Declaration with keyword static
Memory allocated when object is created	Memory is allocated when class is loaded into memory
For each object, separate copy of instance variable will be created	For all objects, only one copy of static variable created
Memory is created in Heap	Memory is created in Method area

- What is a method?

If we write a function in a class, we can call it as method.

- What is the purpose of method?

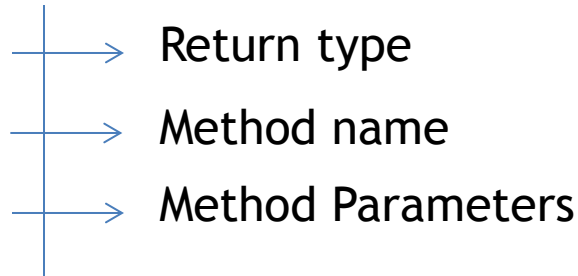
Purpose is to perform some task.

- Syntax:

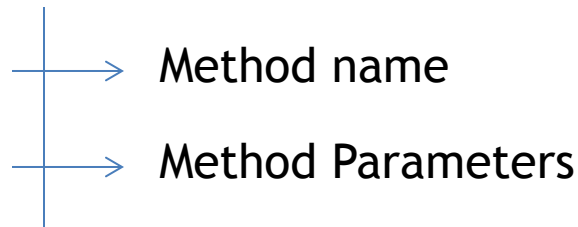
```
returntype methodname()  
{  
    //method body  
}
```

Method [1]

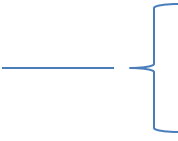
- **Method Header or Method prototype:** It contains



- **Method Signature:** It contains



Method [2]

- **Methods** 
 - 1. Instance Methods.
 - 2. Static Methods.

1. Instance Method:

Defining /writing any method in a class, we can call it as instance method.

- How we can call instance method?

Using object name.

Syntax: objectname.method();
v1.calc();

- To call instance method, we should create an object.

2. Static Method:

Defining/writing a method with the keyword “static”, we can call it as static method.

- How we can call static method?

Using classname

Syntax: classname.methodname();

Ex: Sample.m1();

- Limitation of static method:

```
static void m1()
```

```
{
```

```
    // can't access instance variables/methods
```

```
}
```

- what static method can access?

static variables/methods

local variables

Method [4]

Reason for not accessing instance variables in static methods:

- First when class is loaded, for all static variables/methods/blocks, memory will be allocated in the method area.
- Later, objects/instance will be created in the heap area.
- So, later part [objects/heap area] **can't** be accessed in prior part [method area].
- But prior part [method area] **can** be accessed in later part [objects/heap area].