CONSTRUCTOR

CONSTRUCTOR

Constructor is a special type of non-static method whose name is the same as the class name but it does not have a return type.

Syntax to create the constructor

```
[access_modifier] className([Formal_Arguments])
{
// initialization ;
}
```

CONSTRUCTOR BODY

A constructor body will have the following things :

- Load instructions added by the compiler during compile time.
- Non static initializers of the class.
- Programmer written instructions.

PURPOSE OF THE CONSTRUCTOR

During the execution of the constructor,

- Non-static members of the class will be loaded into the object.
- If there is a non-static initializer in the class they start executing from top to bottom order.
- Programmer written instruction of the constructor gets executed.

NOTE : If the programmer fails to create a constructor then the compiler will add a default constructor.

CLASSIFICATION OF CONSTRUCTOR

Constructors can be classified into two types based on the formal argument,

1. No argument constructor
2. Parameterized constructor

NO ARGUMENT CONSTRUCTOR

A constructor which doesn't have a formal argument is known as a no-argument constructor.

Syntax to create no argument constructor

```
[access modifier] className()
{
//code
}
```

NOTE : If the programmer fails to create a constructor then the compiler implicitly adds a no-argument constructor only.

PARAMETERIZED CONSTRUCTOR

A constructor which has a formal argument is known as parameterized constructor.

Syntax to create parameterized argument constructor

```
[access modifier] className([Formal argument])
  {
     //code
   }
```

PURPOSE OF THE PARAMETERIZED CONSTRUCTOR

Parameterized constructors are used to initialize the variables (non-static) by accepting the data from the constructor in the object creation statement.

LOADING PROCESS OF AN OBJECT

- A new keyword will create a block of memory in a heap area
- Constructor is called.
- During the execution of the constructor,
    a. All the non-static members of the class are loaded into the object.
    b. If there are non-static initializers they are executed from top to bottom order.
    c.. Programmer written instruction of the constructors will be executed.
- The execution of the constructor is completed.
- The object is created successfully.
- The reference of an object is returned by the new keyword.
- These steps are repeated for every object creation.

CONSTRUCTOR OVERLOADING :

- A process of using a number of constructor with the same name but different types of parameters is known as “Constructor overloading”.
- A number of constructors used in a program will have their names same as the class name. Hence, they are overloaded.

DIFFERENCE BETWEEN CONSTRUCTOR AND METHODS

CONSTRUCTOR	METHODS
Constructor name will be the name of class.	We can give any suitable name to Methods.
Constructor don't have return type not even void.	Methods have return type .
Constructor automatically gets invoked.	Methods has to be invoked.

CONSTRUCTOR CHAINING :

- A constructor calling another constructor is known as constructor chaining.
- In java, we can achieve constructor chaining by using two ways

    1. this() (this call statement)
    2. super() (super call statement)

this() :

It is used to call the constructor of the same class from another constructor.

RULE :

- this() can be used only inside the constructor.
- It should always be the first statement in the constructor.
- The recursive call to the constructor is not allowed (Calling by itself).
- If a class has n constructors we can use this statement in n-1 constructors only(at least a constructor should be without this()

NOTE :

If the constructor has this() statement then the compiler doesn’t add load instruction & non-static initializers into the constructor body.