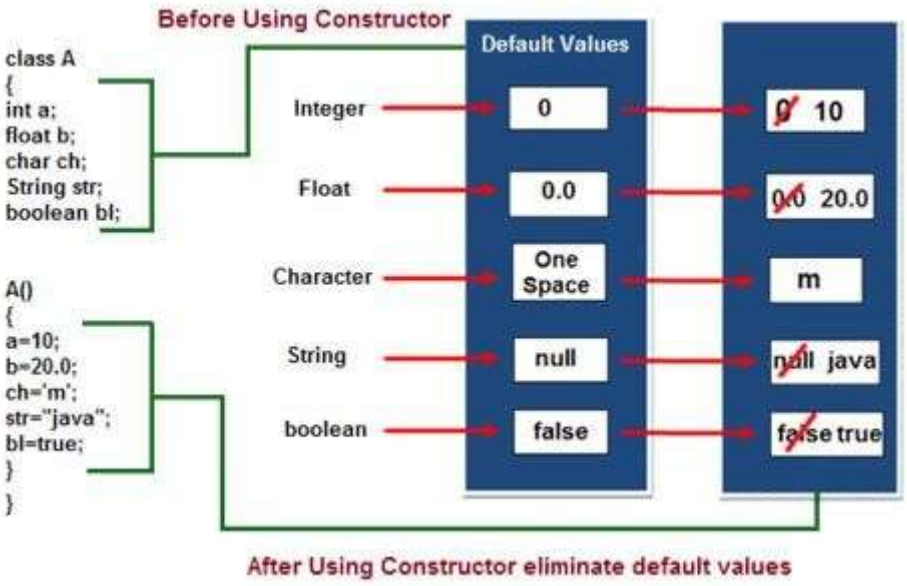
Constructor

Constructor in Java is a special member method which will be called implicitly (automatically) by the JVM whenever an object is created.
Constructors are mainly created for initializing the object. Initialization is a process of assigning user defined values at the time of allocation of memory space.sName()

Syntex
className{
.....
.....
}
Advantages of constructors in Java

- A constructor eliminates placing the default values.
- A constructor eliminates calling the normal or ordinary method implicitly.

How Constructor eliminate default values?
Constructor are mainly used for eliminate default values by user defined values, whenever we create an object of any class then its allocate memory for all the data members and initialize there default values. To eliminate these default values by user defined values we use constructor.



Example

```
class Sum{
 int a,b;Sum()
```

```
{
 a=10;b=20;
    c=a+b;
    System.out.println("Sum: "+c);
    }
    public static void main(String s[])
    {
    Sum s=new Sum();
}
}
```

Output Sum: 30

In above example when we create an object of "Sum" class then constructor of this class call and initialize user defined value in a=10 and b=20. If we cannot create constructor of Sum class then it print " Sum: 0 " because default values of integer is zero.

Rules or properties of a constructor

- Constructor will be called automatically when the object is created.
- Constructor name must be similar to name of the class.
- Constructor should not return any value even void also. If we write the return type for the constructor then that constructor will be treated as ordinary method.
- Constructor definitions should not be static. Because constructors will be called each and every time, whenever an object is created.
- Constructor should not be private provided an object of one class is created in another class (Constructor can be private provided an object of one class created in the same class).
- Constructors will not be inherited from one class to another class (Because every class constructor is created for initializing its own data members).
- The access specifier of the constructor may or may not be private.

1. If the access specifier of the constructor is private then an object of corresponding class can be created in the context of the same class but not in the context of some other classes.
2. If the access specifier of the constructor is not private then an object of corresponding class can be created both in the same class context and in other class context.

Difference between Method and Constructor

	Method	Constructor
1	Method can be any user defined name	1 Constructor must be class name
2	Method should have return type	2 It should not have any return type (even void)
3	Method should be called explicitly either with object reference or class reference	3 It will be called automatically whenever objectis created
4	Method is not provided by compiler in any case.	4 The java compiler provides a default constructor if we do not have any constructor.

Types of constructors

Based on creating objects in Java constructor are classified in two types. They are

Default or no argument Constructor

Parameterized constructor.

- Default Constructor
- A constructor is said to be default constructor if and only if it never take any parameters.
- If any class does not contain at least one user defined constructor than the system will createa default constructor at the time of compilation it is known as system defined default constructor.

Syntax

```
class className
{
..... // Call default constructorclassna{
Block of statements;
  // Initialization
}
.....
}
```

- Note: System defined default constructor is created by java compiler and does not have anystatement in the body part. This constructor will be executed every time whenever an object is created if that class does not contain any user defined constructor.
- Example of default constructor.
- In below example, we are creating the no argument constructor in the Test class. It will beinvoked at the time of object creation.

Example

```
//TestDemo.java class Test
{
int a, b; Test ()
{
System.out.println("I am from default Constructor...");
a=10;
b=20;
System.out.println("Value of a: "+a);
System.out.println("Value of b: "+b);
}
}
class TestDemo
{
public static void main(String [] args)
{
Test t1=new Test ();
}
}
```

Output

```
I am from default
Constructor..Value ofa:10
Value of b: 20
```

Rule-1:
Whenever we create an object only with default constructor, defining the default constructor is optional. If we are not defining default constructor of a class, then JVM will call automatically system defined default constructor. If we define, JVM will call user defined default constructor.

Purpose of default constructor?
Default constructor provides the default values to the object like 0, 0.0, null etc. depending on their type (for integer 0, for string null).
Example of default constructor that displays the default values

```
class Student
{
int roll;
float marks;
String name;
void show()
{
System.out.println("Roll: "+roll);
System.out.println("Marks: "+marks);
System.out.println("Name: "+name);
}
}
class TestDemo
{
public static void main(String [] args)
{
Student s1=new Student();
s1.show();
}
}
```

Output
Roll: 0
Marks: 0.0
Name: null
Explanation: In the above class, we are not creating any constructor so compiler provides a default constructor. Here 0, 0.0 and null values are provided by default constructor.
Parameterized constructor
If any constructor contain list of variable in its signature is known as paremetrized constructor. A parameterized constructor is one which takes some parameters.
Syntax

```
class ClassName
{
.....
ClassName(list of parameters) //parameterized constructor
{
.....
}
.....
}
```

Syntax to call parameterized constructor

```
ClassName  objref=new ClassName(value1, value2,....);
                        OR
new ClassName(value1, value2,....);
```

Example of Parameterized Constructor

```
class Test{
int a, b;
Test(int n1, int n2)
{
System.out.println("I am from Parameterized Constructor...");
a=n1;
b=n2;
System.out.println("Value of a = "+a);
System.out.println("Value of b = "+b);
}
}
class TestDemo1{
public static void main(String k []){
Test t1=new Test(10, 20);
}
}
```

Important points Related to Parameterized Constructor

- Whenever we create an object using parameterized constructor, it must define parameterized constructor otherwise we will get compile time error. Whenever we define the objects with respect to both parameterized constructor and default constructor, it must define both the constructors.

In any class maximum one default constructor but 'n' number of parameterized constructors

Difference between Static and non-Static Variable in Java

The variable of any class are classified into two types;

- Static or class variable
- Non-static or instance variable

Static variable in Java

Memory for static variable is created only one in the program at the time of loading of class. These variables are preceded by static keyword. Static variable can access with class reference.

Non-static variable in Java

Memory for non-static variable is created at the time of creation an object of class. These variables should not be preceded by any static keyword. These variables can access with object reference.

Difference between non-static and static variable

	Non-static variable	Static variable
1	These variable should not be preceded by any static keyword Example: class A { int a; }	These variables are preceded by static keyword. Example class A { static int b; }
2	Memory is allocated for these variable whenever an object is created	Memory is allocated for these variables at the time of loading of the class.
3	Memory is allocated multiple times	Memory is allocated for these variable only once

	whenever a new object is created.	in the program.
4	Non-static variable also known as instance variable while because memory is allocated whenever instance is created.	Memory is allocated at the time of loading of class so that these are also known as class variable.
5	Non-static variable are specific to an object	Static variable are common for every object that means there memory location can be sharable by every object reference or same class.
6	Non-static variable can access with object reference. Syntax obj_ref.variable_name	Static variable can access with class reference. Syntax class_name.variable_name

Note: static variable not only can be access with class reference but also some time it can beaccessed with object reference.

Example of static and non-static variable.

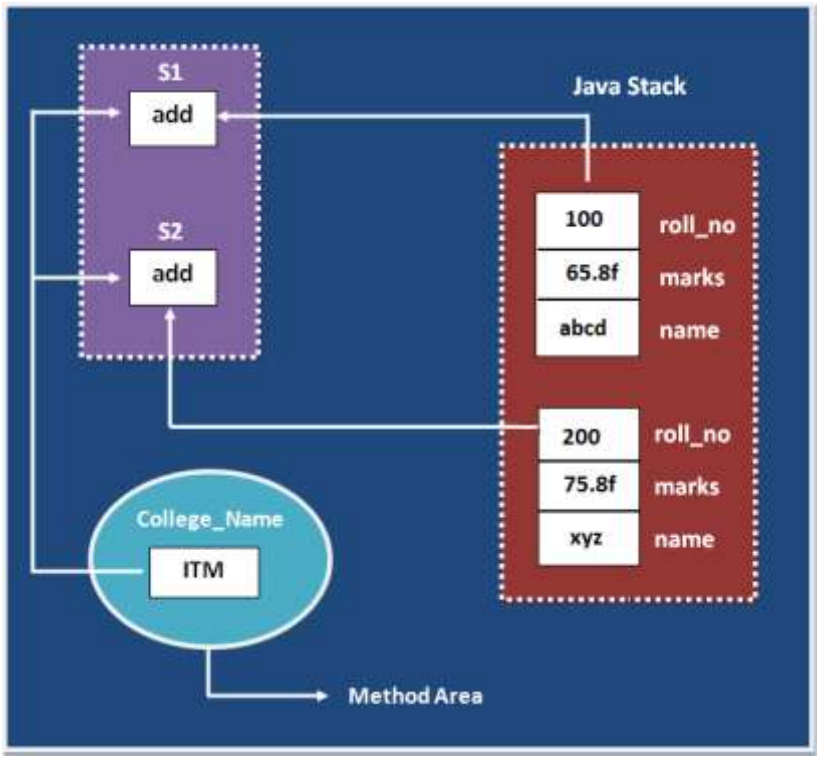
Example

```
class Student
{
int roll_no;
float marks;
String name;
static String College_Name="ITM";
}
class StaticDemo
{
public static void main(String args[])
{
Student s1=new Student();
s1.roll_no=100;
s1.marks=65.8f;
s1.name="abcd";
System.out.println(s1.roll_no);
System.out.println(s1.marks);
System.out.println(s1.name);
System.out.println(Student.College_Name);
//or System.out.println(s1. College_Name); but first is use in real time.
Student  s2=new  Student();
s2.roll_no=200;
s2.marks=75.8f;
s2.name="zyx";
System.out.println(s2.roll_no);
System.out.println(s2.marks);
System.out.println(s2.name);
System.out.println(Student.College_Name);
}
}
```

Output

```
100
65.8
abcd
```

ITM
200
75.8
zyx
ITM



Note: In the above example College_Name variable is commonly sharable by both S1 and S2 objects.

Understand static and non-static variable using counter

Program of counter without static variable

In this example, we have created an instance variable named count which is incremented in the constructor. Since instance variable gets the memory at the time of object creation, each object will have the copy of the instance variable, if it is incremented, it won't reflect to other objects. So each objects will have the value 1 in the count variable.

Example

```
class Counter
{
int count=0;//will get memory when instance is created
Counter()
{
count++;
System.out.println(count);
}
public static void main(String args[])
{
Counter c1=new Counter();
Counter c2=new Counter();
Counter c3=new Counter();
}
}
```

Output
1
1
1
Program of counter by static variable
As we have mentioned above, static variable will get the memory only once, if any object changes the value of the static variable, it will retain its value.
Example
```
class Counter
{
static int count=0; //will get memory only once
Counter()
{
count++;
System.out.println(count);
}
public static void main(String args[])
{
Counter c1=new Counter();
Counter c2=new Counter();
Counter c3=new Counter();
}
}
```
Output
1
2
3
Difference between Static and non-static method in Java
In case of non-static method memory is allocated multiple times whenever method is calling. But memory for static method is allocated only once at the time of class loading. Method of a class can be declared in two different ways

- Non-static methods
- Static methods

Difference between non-static and static Method

	Non-Static method	Static method
1	These method never be preceded by static keyword Example: void fun1() { .... }	These method always preceded by static keyword Example: static void fun2() { .... }
2	Memory is allocated multiple time whenever method is calling.	Memory is allocated only once at the time of class loading.
3	It is specific to an object so that these are also known as instance method.	These are common to every object so that it is also known as member method or class method.
4	These methods always access with object reference	These property always access with class reference

	Syntax: Objref.methodname();	Syntax: className.methodname();
5	If any method wants to be execute multiple time that can be declare as non static.	If any method wants to be execute only once in the program that can be declare as static .

Note: In some cases static methods not only can access with class reference but also can access with object reference.

Example of Static and non-Static Method

Example

```
class A
{
void fun1()
{
System.out.println("Hello I am Non-Static");
}
static void fun2()
{
System.out.println("Hello I am Static");
}
}
class Person
{
public static void main(String args[])
{
A  oa=new  A();
  oa.fun1(); // non static method
  A.fun2();  // static method
}
}
```

Output

Hello I am Non-Static

Hello I am Static

Following table represent how the static and non-static properties are accessed in the different static or non-static method of same class or other class.

	within non-static method of same class	within static method of same class	within non-static method of other class	within static method of other class
Non-Static Method or Variable	Access directly	Access with object reference	Access with object reference	Access with object reference
Static Method or Variable	Access directly	Access directly	Access with object reference	Access with object reference

Program to accessing static and non-static properties.

Example

```
class A
{
```

```
int y;
void f2()
{
System.out.println("Hello f2()");
}
}
class B
{
int z;
void f3()
{
System.out.println("Hello f3()");
A a1=new A();
a1.f2();
}
}
class Sdemo
{
static int x;
static void f1()
{
System.out.println("Hello f1()");
}
public static void main(String[] args)
{
x=10;
System.out.println("x="+x);
f1();
System.out.println("Hello main");
B b1=new B();
b1.f3();
}
}
```