Constructor In Java

Constructor :

In Java, constructor is a block of codes similar to method. It is called when an instance of object is created and memory is allocated for the object.

It is a special type of method which is used to initialize the object.

When is a constructor called ?

Everytime an object is created using new() keyword, atleast one constructor is called. It is called a default constructor.

Constructor In Java

Rules for creating java constructor :

There are basically two rules defined for the constructor.

- Constructor name must be same as its class name
- Constructor must have no explicit return type

Types of java constructors

There are two types of constructors in java:

- Default constructor (no-arg constructor)
- Parameterized constructor

Constructor In Java

Java Default Constructor

A constructor is called "Default Constructor" when it doesn't have any parameter.

Syntax of default constructor:

```
<class_name>()
 {

 }
```

Constructor In Java

Example of default constructor :

In this example, we are creating the no-arg constructor in the Bike class. It will be invoked at the time of object creation.

```
  class Bike1
{
  Bike1()
 {System.out.println("Bike is created");}
  public static void main(String args[])
  {
  Bike1 b=new Bike1();
  }
}
```

Constructor In Java

If there is no constructor in a class, compiler automatically creates a default constructor.

What is the purpose of default constructor ?

Default constructor is used to provide the default values to the object like 0, null etc. depending on the type.

Constructor In Java

Example of default constructor that displays the default values

```
class Student3{
int id;
String name;

void display(){System.out.println(id+" "+name);}

public static void main(String args[]){
Student3 s1=new Student3();
Student3 s2=new Student3();
s1.display();
s2.display();
}  }
```

Constructor In Java

Java parameterized constructor

A constructor which has a specific number of parameters is called parameterized constructor.

Why use parameterized constructor ?

Parameterized constructor is used to provide different values to the distinct objects.

Constructor In Java

```
class Student{
    int id;
    String name;

    Student(int i,String n){
    id = i;
    name = n;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    Student s1 = new Student(111,"Karan");
    Student s2 = new Student(222,"Aryan");
    s1.display();
    s2.display();
   }  }
```

Constructor Overloading in Java :

In Java, a constructor is just like a method but without return type. It can also be overloaded like Java methods.

Constructor overloading in Java is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task. They are differentiated by the compiler by the number of parameters in the list and their types.

Constructor In Java

```
class Student{
    int id;
    String name;
    int age;
    Student(int i,String n){
    id = i;
    name = n;
    }
    Student(int i,String n,int a){
    id = i;
    name = n;
    age=a;
    }
```

Constructor In Java

```
void display(){System.out.println(id+" "+name+" "+age);}

    public static void main(String args[]){
    Student s1 = new Student(111,"Karan");
    Student s2 = new Student(222,"Aryan",25);
    s1.display();
    s2.display();
   }
}
```

Constructor In Java

Difference between constructor and method in java :

Java Constructor	Java Method
Constructor is used to initialize the state of an object.	Method is used to expose behaviour of an object.
Constructor must not have return type.	Method must have return type.
Constructor is invoked implicitly.	Method is invoked explicitly.
The java compiler provides a default constructor if you don't have any constructor.	Method is not provided by compiler in any case.
Constructor name must be same as the class name.	Method name may or may not be same as class name.

Java Copy Constructor

There is no copy constructor in java. But, we can copy the values of one object to another like copy constructor in C++.

There are many ways to copy the values of one object into another in java. They are:

- By constructor
- By assigning the values of one object into another
- By clone() method of Object class

Constructor In Java

```
class Student{
    int id;
    String name;
    Student(int i,String n)
   {
    id = i;
    name = n;
    }

    Student(Student s)
    {
    id = s.id;
    name =s.name;
    }
```

Constructor In Java

```
void display()
{
  System.out.println(id+" "+name);
}

    public static void main(String args[])
  {
    Student s1 = new Student(111,"Karan");
    Student s2 = new Student(s1);
    s1.display();
    s2.display();
   }
}
```

Copying values without constructor

```
class Student{
    int id;
    String name;
    Student(int i,String n)
    {
    id = i;
    name = n;
    }
    Student(){}
    void display()
    {System.out.println(id+" "+name);}
```

Constructor In Java

```
 public static void main(String args[]){
    Student s1 = new Student(111,"Karan");
    Student s2 = new Student();
    s2.id=s1.id;
    s2.name=s1.name;
    s1.display();
    s2.display();
   }
}
```

Constructor In Java

Does constructor return any value?

yes, that is current class instance (You cannot use return type yet it returns a value).

Can constructor perform other tasks instead of initialization?

Yes, like object creation, starting a thread, calling method etc. You can perform any operation in the constructor as you perform in the method.

Java static keyword

The static keyword in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class.

The static can be:

- **variable (also known as class variable)**
- **method (also known as class method)**
- **block**
- **nested class**

Java static keyword

1) Java static variable

If you declare any variable as static, it is known static variable.

A. The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees,college name of students etc.

B. The static variable gets memory only once in class area at the time of class loading.

Java static keyword

Advantage of static variable :

It makes your program memory efficient (i.e it saves memory).

Understanding problem without static variable

```
class Student{
    int rollno;
    String name;
    String      college="KIIT";
}
```

Java static keyword

```
//Program of static variable

public class Student{
   int rollno;
   String name;
   static String college ="KIIT";

   Student(int r,String n){
   rollno = r;
   name = n;
   }
  void display (){System.out.println(rollno+" "+name+"
"+college);}
```

Java static keyword

```
public static void main(String args[]){
   Student s1 = new Student(111,"Karan");
   Student s2 = new Student(222,"Aryan");

   s1.display();
   s2.display();
   }
  }
```

Java static keyword

```
//Program of counter without static variable
public 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();
  }
}
```

Java static keyword

```
//Program of counter by static variable
 public class Counter{
static int count=0;//will get memory only once and retain its value
Counter(){
count++;
System.out.println(count);
}
public static void main(String args[]){

Counter c1=new Counter();
Counter c2=new Counter();
Counter c3=new Counter();
  }
}
```

Java static keyword

2) Java static method

If you apply static keyword with any method, it is known as static method.

i. A static method belongs to the class rather than object of a class.

ii. A static method can be invoked without the need for creating an instance of a class.

iii. static method can access static data member and can change the value of it.

Java static keyword

```
//Program of changing the common property of all objects(static field).
public class Student{
    int rollno;
    String name;
    static String college = "KIIT";

    static void change(){
    college = "KISS";
    }
    Student(int r, String n){
    rollno = r;
    name = n;
    }
```

Java static keyword

//Program of changing the common property of all objects(static field).

```
void display (){System.out.println(rollno+" "+name+"
"+college);}
public static void main(String args[]){
  Student.change();

  Student s1 = new Student(111,"Karan");
  Student s2 = new Student(222,"Aryan");
  Student s3 = new Student(333,"Laxman");
  s1.display();
  s2.display();
  s3.display();
  }}
```

Java static keyword

//Program to get cube of a given number by static method

```
public class Calculate{
  static int cube(int x){
  return x*x*x;
  }

  public static void main(String args[]){
  int result=Calculate.cube(5);
  System.out.println(result);
  }
}
```

Java static keyword

Restrictions for static method

There are two main restrictions for the static method. They are:

- **The static method can not use non static data member or call non-static method directly.**
- **this and super cannot be used in static context.**

Java static keyword

Restrictions for static method

```
class A{
 int a=40;//non static

 public static void main(String args[]){
  System.out.println(a);
 }
}
```

Java static keyword

why java main method is static ?

Because object is not required to call static method if it were non-static method, jvm create object first then call main() method that will lead the problem of extra memory allocation.

Java static keyword

3) Java static block

- Is used to initialize the static data member.
- It is executed before main method at the time of classloading.

Example of static block

```
public class A2{
  static{
  System.out.println("static block is invoked");}
  public static void main(String args[]){
   System.out.println("Hello main");
  } }
```

Java static keyword

Can we execute a program without main() method ?

Yes, one of the way is static block but in previous version of JDK not in JDK 7.

```
public class A3{
  static{
  System.out.println("static block is invoked");
  System.exit(0);
  }
}
```

- **Output:static block is invoked (if not JDK7)**

Java static keyword

Can we execute a program without main() method ?

In JDK7 and above, output will be:

Output:Error: Main method not found in class A3, please define the main method as:
public static void main(String[] args)

Java this keyword

Usage of java this keyword

- this can be used to refer current class instance variable.
- this can be used to invoke current class method (implicitly)
- this() can be used to invoke current class constructor.
- this can be passed as an argument in the method call.
- this can be passed as argument in the constructor call.
- this can be used to return the current class instance from the method.

Java this keyword

1) this: to refer current class instance variable

The this keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

Java this keyword

Understanding the problem without this keyword

```
class Student{
int rollno;
String name;
float fee;
Student(int rollno,String name,float fee){
rollno=rollno;
name=name;
fee=fee;
}
void display(){System.out.println(rollno+" "+name+" "+fee);}
}
```

Java this keyword

Understanding the problem without this keyword

```
public class TestThis1{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

Output:

```
0 null 0.0
0 null 0.0
```

Java this keyword

Solution of the above problem by this keyword

```
class Student{
int rollno;
String name;
float fee;
Student(int rollno,String name,float fee){
this.rollno=rollno;
this.name=name;
this.fee=fee;
}
void display(){System.out.println(rollno+" "+name+" "+fee);}
}
```

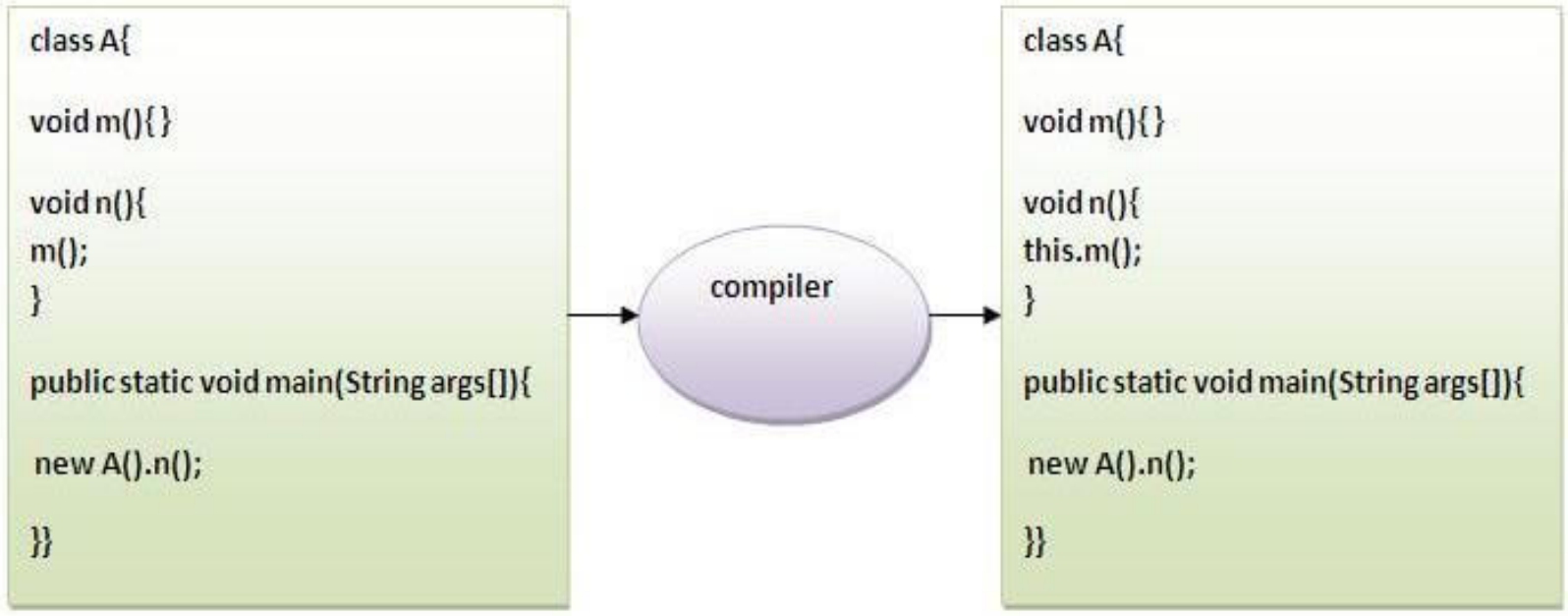
Java this keyword

```
public class TestThis{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

Java this keyword

2) this: to invoke current class method

You may invoke the method of the current class by using the this keyword. If you don't use the this keyword, compiler automatically adds this keyword while invoking the method.



Java this keyword

```
class A{
void m(){System.out.println("hello m");}
void n(){
System.out.println("hello n");
//m();//same as this.m()
this.m();
}
}
class TestThis{
public static void main(String args[]){
A a=new A();
a.n();
}}
```

3) this() : to invoke current class constructor

The this() constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.

Java this keyword

Calling default constructor from parameterized constructor:

```
class A{
A(){System.out.println("hello a");}
A(int x){
this();
System.out.println(x);
}
}
public class TestThis{
public static void main(String args[]){
A a=new A(10);
}}
```

Java this keyword

Calling parameterized constructor from default constructor:

```
class A{
A(){
this(5);
System.out.println("hello a");
}
A(int x){
System.out.println(x);
}  }
public class TestThis{
public static void main(String args[]){
A a=new A();
}}
```

Java this keyword

Constructor chaining:

```
class Student{
int rollno;
String name,course;
float fee;
Student(int rollno,String name,String course){
this.rollno=rollno;
this.name=name;
this.course=course;
}
Student(int rollno,String name,String course,float fee){
this(rollno,name,course);//reusing constructor
this.fee=fee;
}
```

Java this keyword

Constructor chaining:

```
void display(){System.out.println(rollno+" "+name+"
"+course+" "+fee);}
}
public class TestThis{
public static void main(String args[]){
Student s1=new Student(111,"ankit","java");
Student s2=new Student(112,"sumit","java",6000f);
s1.display();
s2.display();
}}
```

Java this keyword

Rule: Call to this() must be the first statement in constructor.

```
class Student{
int rollno;
String name,course;
float fee;
Student(int rollno,String name,String course){
this.rollno=rollno;
this.name=name;
this.course=course;
}
Student(int rollno,String name,String course,float fee){
this.fee=fee;
this(rollno,name,course);//C.T.Error
}
```

Java this keyword

4) this: to pass as an argument in the method

```
public class S2{
  void m(S2 obj){
  System.out.println("method is invoked");
  }
  void p(){
  m(this);
  }
  public static void main(String args[]){
  S2 s1 = new S2();
  s1.p();
  }
}
```

Java this keyword

5) this: to pass as argument in the constructor call

```
class B{
  A obj;
  B(A obj){
    this.obj=obj;
  }
  void display(){
    System.out.println(obj.data);//using data member of A class
  }
}
```

Java this keyword

5) this: to pass as argument in the constructor call

```
public class A{
  int data=10;
  A(){
   B b=new B(this);
   b.display();
  }
  public static void main(String args[]){
   A a=new A();
  }  }
```

Java this keyword

6) this keyword can be used to return current class instance

```
class A{
A getA(){
return this;
}
void msg(){System.out.println("Hello java");}
}
class Test1{
public static void main(String args[]){
new A().getA().msg();
}
}
```

Proving this keyword

Let's prove that this keyword refers to the current class instance variable. In this program, we are printing the reference variable and this, output of both variables are same.

Proving this keyword

```
class A5{
void m(){
System.out.println(this);//prints same reference ID
}
public static void main(String args[]){
A5 obj=new A5();
System.out.println(obj);//prints the reference ID
obj.m();
}
}
```