

## Basics of Java

### OOPs Concepts

Advantage of OOPs

Naming Convention

Object and Class

Method Overloading

Constructor

static variable, method and block

this keyword

Inheritance (IS-A)

Aggregation (HAS-A)

Method Overriding

Covariant Return Type

super keyword

Instance Initializer block

final keyword

Abstract class

Interface

Runtime Polymorphism

static and Dynamic binding

Downcasting with instanceof operator

Package

Access Modifiers

Encapsulation

Object class

Object Cloning

Java Array

Call By Value

strictfp keyword

API Document

Command Line

Argument

String Handling

Exception Handling

Nested Classes

Multithreading

Synchronization

I/O

Serialization

Networking

AWT

Event Handling

Swing

LayoutManager

Applet

Reflection API

Collection

JDBC

Java New Features

RMI

Internationalization

## Instance initializer block:

Instance Initializer block is used to initialize the instance data member. It runs each time when an object of the class is created.

The initialization of the instance variable can be directly but there can be performed extra operations while initializing the instance variable in the instance initializer block.

[Instance initializer block](#)

[Example of Instance initializer block](#)

[What is invoked firstly instance initializer block or constructor?](#)

[Rules for instance initializer block](#)

[Program of instance initializer block that is invoked after super\(\)](#)

**Que) What is the use of instance initializer**

**block while we can directly assign a value in instance data member? For example:**

```
class Bike{
    int speed=100;
}
```

## Why use instance initializer block?

Suppose I have to perform some operations while assigning value to instance data member e.g. a for loop to fill a complex array or error handling etc.

## Example of Instance initializer block

Let's see the simple example of instance initializer block that performs initialization.

```
<b><i>Program of instance initializer block that initializes values to the instance variable</i></b>

class Bike{
    int speed;

    Bike(){System.out.println("speed is "+speed);}

    {speed=100;}

    public static void main(String args[]){
        Bike b1=new Bike();
        Bike b2=new Bike();
    }
}
```

**Output:**speed is 100  
speed is 100

There are three places in Java where you can perform operations:

1. method
2. constructor
3. block

## What is invoked firstly instance initializer block or constructor?

```
<b><i>Program of instance initializer block</i></b>

class Bike{
    int speed;
```



```
Bike(){System.out.println("constructor is invoked");}

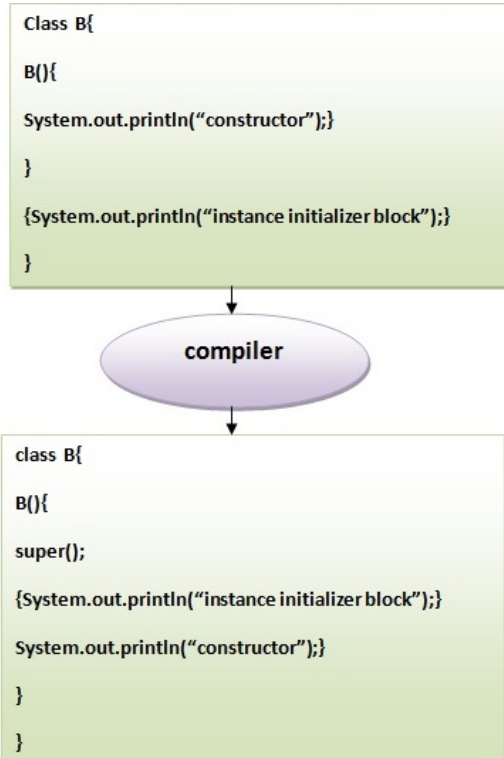
{System.out.println("instance initializer block invoked");}

public static void main(String args[]){
    Bike b1=new Bike();
    Bike b2=new Bike();
}
```

**Output:**instance initializer block invoked  
constructor is invoked  
instance initializer block invoked  
constructor is invoked

In the above example, it seems that instance initializer block is firstly invoked but NO. Instance initializer block is invoked at the time of object creation. The java compiler copies the instance initializer block in the constructor after the first statement super(). So firstly, constructor is invoked. Let's understand it by the figure given below:

**Note:** The java compiler copies the code of instance initializer block in every constructor.



## Rules for instance initializer block :

There are mainly three rules for the instance initializer block. They are as follows:

1. The instance initializer block is created when instance of the class is created.
2. The instance initializer block is invoked after the parent class constructor is invoked (i.e. after super() constructor call).
3. The instance initializer block comes in the order in which they appear.

## Program of instance initializer block that is invoked after super()

<b><i>//Program of instance initializer block that is invoked after super()</i></b>

```
class A{

A(){
    System.out.println("parent class constructor invoked");
}
```

```

}

class B extends A{

B(){
    super();
    System.out.println("child class constructor invoked");
}

{System.out.println("instance initializer block is invoked");}

public static void main(String args[]){
    B b=new B();
}
}

```

**Output:**parent class constructor invoked  
instance initializer block is invoked  
child class constructor invoked

**<b><i>Another example of instance initializer block that is invoked after super()</i></b>**

```

class A{

A(){
    System.out.println("parent class constructor invoked");
}

}

class B extends A{

B(){
    super();

    System.out.println("child class constructor invoked");
}

B(int a){
    super();

    System.out.println("child class constructor invoked "+a);
}

{System.out.println("instance initializer block is invoked");}

public static void main(String args[]){
    B b1=new B();
    B b2=new B(10);
}
}

```

**Output:**parent class constructor invoked  
instance initializer block is invoked  
child class constructor invoked  
parent class constructor invoked  
instance initializer block is invoked  
child class constructor invoked 10

[<<prev](#)

[next>>](#)