Instance initializer block

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

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

**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.   1. **class** Bike8{ 2. **int** speed; 4. Bike8(){System.out.println("constructor is invoked");} 6. {System.out.println("instance initializer block invoked");} 8. **public** **static** **void** main(String args[]){ 9. Bike8 b1=**new** Bike8(); 10. Bike8 b2=**new** Bike8(); 11. } 12. }     In the above example, it seems that instance initializer block is firstly invoked but NO. Instance intializer 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: **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. |