HowToDoInJava

Java Instance Initializer Blocks

Last Updated: January 30, 2022 By: Lokesh Gupta Java Object Oriented Programming Java OOP

Java instance initializers are code blocks which are executed before the constructor code is executed. These initializers run everytime we create a new object.

1. Instance initializer syntax

An instance initializer block is created with **curly braces**. The object initialization statements are written inside the braces.

```
Syntax

public class DemoClass {

//This is initializer block 1
{
//statements
}

//This is initializer block 2
{
//statements
}
}
```

2. Java instance initializer features

The instance initializers have following features.

- We can define **multiple initializers** in a class.
- All initializers execute in sequence in order they appear in class body.
- Initializers run after the parent class constructor has been invoked and before executing child class constructor. Please note that Java inserts the default constructor of parent class `super()`, if we do not explicitly provide the constructor as the **first statement** in child class's constructor.
- After all the initializers have executed, constructor's statements are executed.
- We can use call the constructors of this class and parent classes inside initializers.

3. Java instance initialization sequence flow

Based on above given features, let's draw an outline how the instance initialization of an object flows.

1. Child class constructor is invoked.
2. Child class constructor has first statement as **super()** (or provided explicit constructor) so parent class constructor is invoked.
3. Parent class's initializers are executed in sequence of their appearance.
4. Parent class constructor statements are executed.
5. Child class's initializers are executed in sequence of their appearance.
6. Child class constructor statements are executed.

4. Java instance initializer example

Let's quickly see an example to demo above theory.

```
ParentClass.java

public class ParentClass
{
```

```
public ParentClass()
{
System.out.println("In ParentClass Constructor");
}

//Instance Initializer
{
System.out.println("In ParentClass Instance Initializer");
}
}
```

```
ChildClass.java

public class ChildClass extends ParentClass
{
public ChildClass()
{
super();  //If not provided, JVM will insert it
System.out.println("In ChildClass Constructor");
}

//Instance Initializer 1
{
System.out.println("In ChildClass Instance Initializer 1");
}

//Instance Initializer 2
{
System.out.println("In ChildClass Instance Initializer 2");
}
}
```

```
Main.java

public class Main
{
public static void main(String[] args)
{
ChildClass childObj = new ChildClass();
}
}
```

Program output.

```
Console
```

```
In ParentClass Instance Initializer
In ParentClass Constructor
In ChildClass Instance Initializer 1
In ChildClass Instance Initializer 2
In ChildClass Constructor
```

Happy Learning !!

Was this post helpful?

Let us know if you liked the post. That's the only way we can improve.

Yes

No

Recommended Reading:

1. Checked exceptions thrown in initializer blocks can be declared by the constructors
2. How to Create and SSH Connect to AWS EC2 Instance
3. Java puzzle – Why try-catch-finally blocks require braces?
4. Java try catch finally Blocks
5. Java – Text blocks and best practices
6. Interface vs Abstract Class in Java
7. Encapsulation vs Abstraction in Java
8. Overloading vs Overriding in Java
9. Java Access Modifiers
0. Constructors in Java

Leave a Comment

Name *

Email *

Website

☐ Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search …

HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise
- Guest Posts

Blogs

REST API Tutorial