Constructor Chaining in Java

Description: Constructor chaining is the process of calling one constructor from another constructor. The following is a discussion of how it can be done specifically with code examples.

What is Constructor Chaining?

A constructor in Java is a specific method used in object creation of a class. The constructor is invoked every time an object of the class is created. It can be used to assign values to the properties of the object at creation time. There can be multiple constructors in a Java class with different parameter lists.

Constructor chaining is used to invoke different implementations of constructors of the same class/parent class at the object creation time.

How to call the constructors?

There are two ways of chaining constructors as follows.

- using this() keyword – to call constructors of the same class
- using super() keyword – to call constructors of the parent class

This is explained in the following examples.

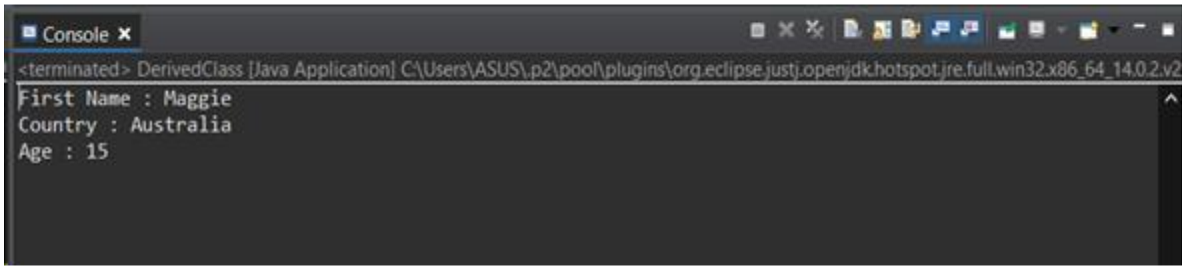
constructor chaining example 1 – Constructors are chained using this() keyword

We have declared four constructors for the DerivedClass. One with no arguments and the other three with different arguments. Inside each constructor this() keyword is used to call one of the other constructor of the same class.

```
package com.tutorialwriting.constchaining;

public class DerivedClass{

    String firstName;
    String country;
    int age;

```

```java
    public DerivedClass() {
        // calling one argument constructor
        this("Maggie");
    }

    public DerivedClass(String firstName) {
        // calling two argument constructor
        this(firstName, 15);
    }

    public DerivedClass(String firstName, int age) {
        // calling three argument constructor
        this(firstName, age, "Australia");
    }

    public DerivedClass(String firstName, int age, String country) {
        this.firstName = firstName;
        this.age = age;
        this.country = country;
    }

    void displayValues() {
        System.out.println("First Name : " + firstName);
        System.out.println("Country : " + country);
        System.out.println("Age : " + age);
    }

    public static void main(String args[]) {
        DerivedClass object = new DerivedClass();
        object.displayValues();
    }
}
```

The output of the execution



```
<terminated> DerivedClass [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v2
First Name : Maggie
Country : Australia
Age : 15
```

constructor chaining example 2 – Constructors are chained using super() keyword

Here, the child class calls the constructors of the parent class using super() keyword. The BaseClass has three constructors. The no argument constructor calls one of the three argument constructor of the BaseClass using this().

```
package com.tutorialwriting.constchaining;

public class BaseClass {

    public BaseClass() {
        //calling a three argument constructor of the same class
        this("Male", "English", "1989/11/10");
        System.out.println("I'm executed third!!!");
    }

    public BaseClass(String firstName, String surname, int idNo) {
        System.out.println("I'm executed first!");
        System.out.println("First name : " + firstName);
        System.out.println("Surname : " + surname);
        System.out.println("ID Number : " + idNo);
    }

    public BaseClass(String gender, String nationality, String birthDate)
{
        System.out.println("I'm executed second!!");
        System.out.println("Gender : " + gender);
        System.out.println("Nationality : " + nationality);
        System.out.println("Birth Date : " + birthDate);
    }


}
```

The DerivedClass has two constructors each calling different constructor of the super class using super().

```
package com.tutorialwriting.constchaining;

public class DerivedClass extends BaseClass {

    public DerivedClass() {
        //calling no argument constructor of the super class
        super();
    }

    public DerivedClass(String firstName, String surname, int idNo) {
        //calling three argument constructor of the super class
        super(firstName, surname, idNo);
    }

    public static void main(String args[]) {
        DerivedClass object2 = new DerivedClass("Paul", "Wilson", 123456);
        DerivedClass object1 = new DerivedClass();
    }
}
```

Implicit vs Explicit constructor calling

Java has two different ways of calling constructors: Implicit calling and Explicit calling.

- Explicit calling refers to calling constructors explicitly in the code using this() or super().
- Implicit calling refers to the calling of the no argument constructor of the super class implicitly at the absence of such an explicit call from the child class constructor. In other words, the compiler adds the super() call as the first line of any of the constructors of the child classes if the programmer explicitly does not call super() in the code.

Why do we need constructor chaining?

There are several different purposes of having a constructor chain in Java as listed below.

- It is a way to access properties of other constructors or properties of parent classes.
- While calling other constructors only one object is being used which is the current instance of the class. The initialization happens in one place but we have the privilege of calling different constructor implementations through a chain. This helps greatly in memory management and code maintenance.

Conclusion

In this tutorial we discussed constructor chaining in Java. Constructors are method-like code segments that are invoked while creating objects. A Java class can have any number of constructors with different parameter lists. Constructor chaining is a handy way of handling different initializations with a one instance of a class. Some important points to be noted from this tutorial are as listed below.

- If the programmer does not add it explicitly to the code, the compiler adds a public no argument constructor to the Java class. This is called the default no argument constructor.
- this() and super() should be written as the first line of the constructor.
- this() is used to call constructors of the same class while super() is used to call constructors of the immediate super class.
- there should be at least one constructor within the class that does not contain this() keyword.
- If not added explicitly, the compiler adds a no argument super() call to every child class constructor. This will help the instantiation of classes correctly.