Constructor Chaining

Constructor calling

Process of one constructor calling another constructor is known as constructor calling. Constructor can be called either by new or constructor calling statements. There are two constructor calling statements:
1. Super();
2. This();

Super calling statement

Super calling statement is used to call the constructor of super class.

This calling statement

This calling statement is ued to call the constructor of the same class.

Rules to declare constructor calling:

1. Constructor calling statement must be the first statement under constructor body.
2. They cannot declare multiple constructor calling statements in one constructor.

```
class A
{
A(double d)
{
}
}

class B extends A
{
B(int x)
{
System.out.println("Value of x is:"+x);
This(14); //error statement should be declared in the first line
}
}

Class A
{
A(double d)
{
}
}
class B extends A
{
B(int x)
{
This(14); //error within same constructor this and super cannot be declared
Super(1.4);
}
}
```

Constructor chaining

Process of one constructor calling another constructor and called constructor calling some other constructor is known as constructor chaining.

Super class constructor cannot be inherited to the sub-class. Super class constructor can be called from sub-class constructor.

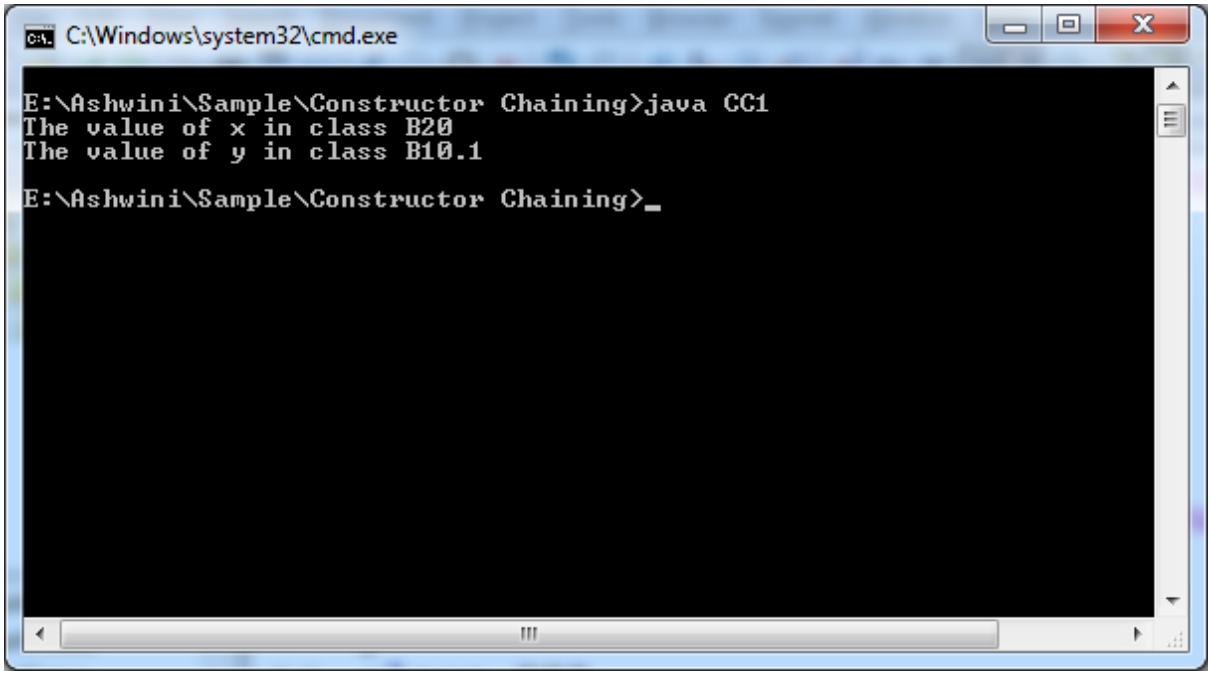
If there is a no argument constructor in superclass, it can be called either implicitly or explicitly.

If there is a parameterised constructor in super class, then it must be called explicitly.

Constructor chaining examples:

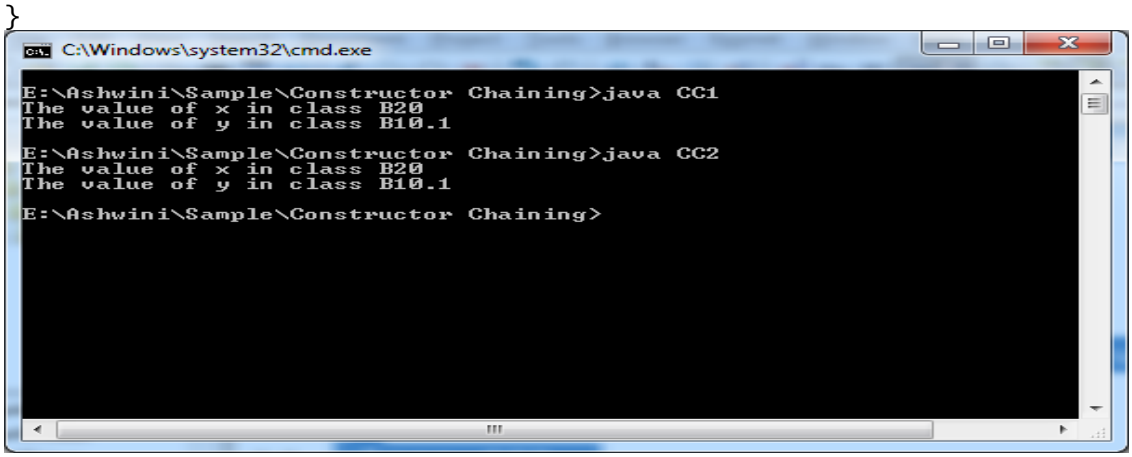
1. Program to call this and super constructor

```
class A
{
	A(int x)
	{
		System.out.println("The value of x in class A"+x);
	}
}
class B extends A
{
	B(int x)
	{
		super(10);
		System.out.println("The value of x in class B"+x);
	}
	B(double y)
	{
		this(20);
		System.out.println("The value of y in class B"+y);
	}
}
class CC1
{
	public static void main(String[] args)
	{
		B b=new B(10.1);
	}
}
```



2. Program to call this constructor

```
class B
{
        B(int x)
        {
                System.out.println("The value of x in class B"+x);
        }

        B(double y)
        {
                this(20);
                System.out.println("The value of y in class B"+y);
        }
}
class CC2
{
        public static void main(String[] args)
        {
                B b=new B(10.1);
        }
}
```



3. Program to cal super constructor

```
class A
{
	A(double d)
	{
		System.out.println("The value of y in class A "+d);
	}
}

class B extends A
{
	B(int x)
	{
		super(1.2);
		System.out.println("The value of x in class B "+x);
	}
}
class CC3
{
	public static void main(String[] args)
	{
		B b=new B(10);
	}
}
```

4. Implicit constructor calling Super constructor.

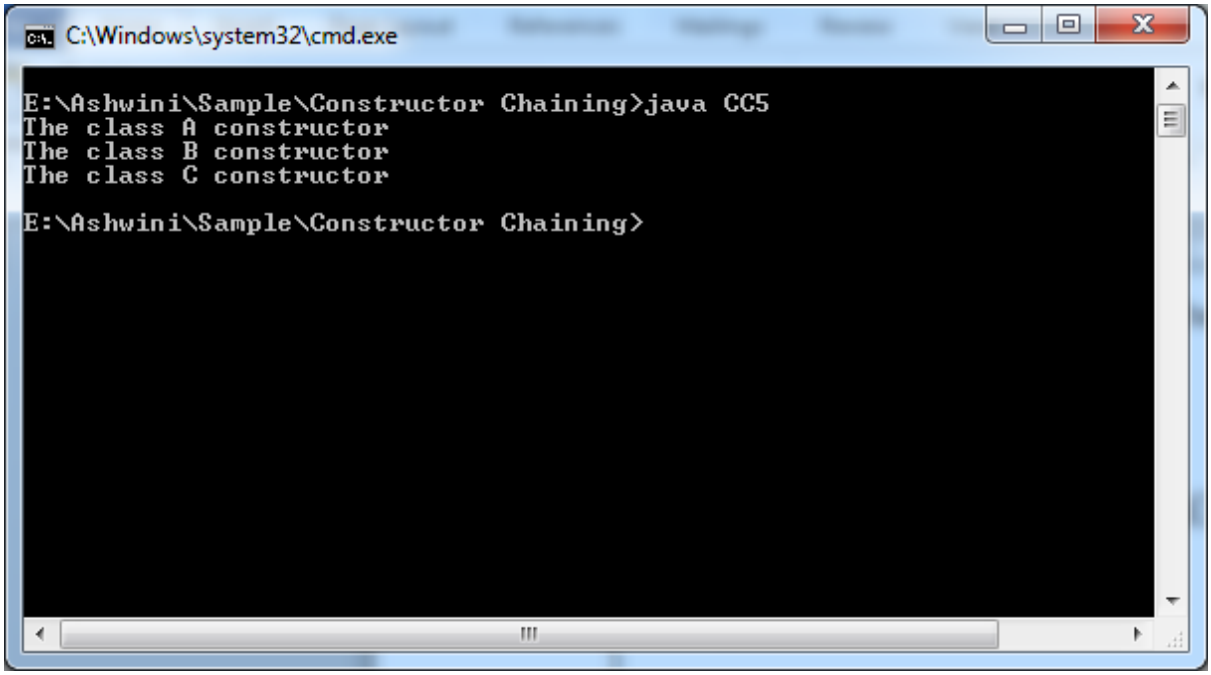
```
class A
{
	A()
	{
		System.out.println("The class A constructor");
	}
}

class B extends A
{
	B(int x)
	{
		System.out.println("The value of x in class B "+x);
	}
}
class CC4
{
	public static void main(String[] args)
	{
		B b=new B(10);
	}
}
```

5. Explicit constructor calling

```
class A
{
	A(double d)
	{
		System.out.println("The value of y in class A "+d);
	}
}

class B extends A
{
	B(int x)
	{
		super(1.2);
		System.out.println("The value of x in class B "+x);
	}
}
class CC3
{
	public static void main(String[] args)
	{
		B b=new B(10);
	}
}
```

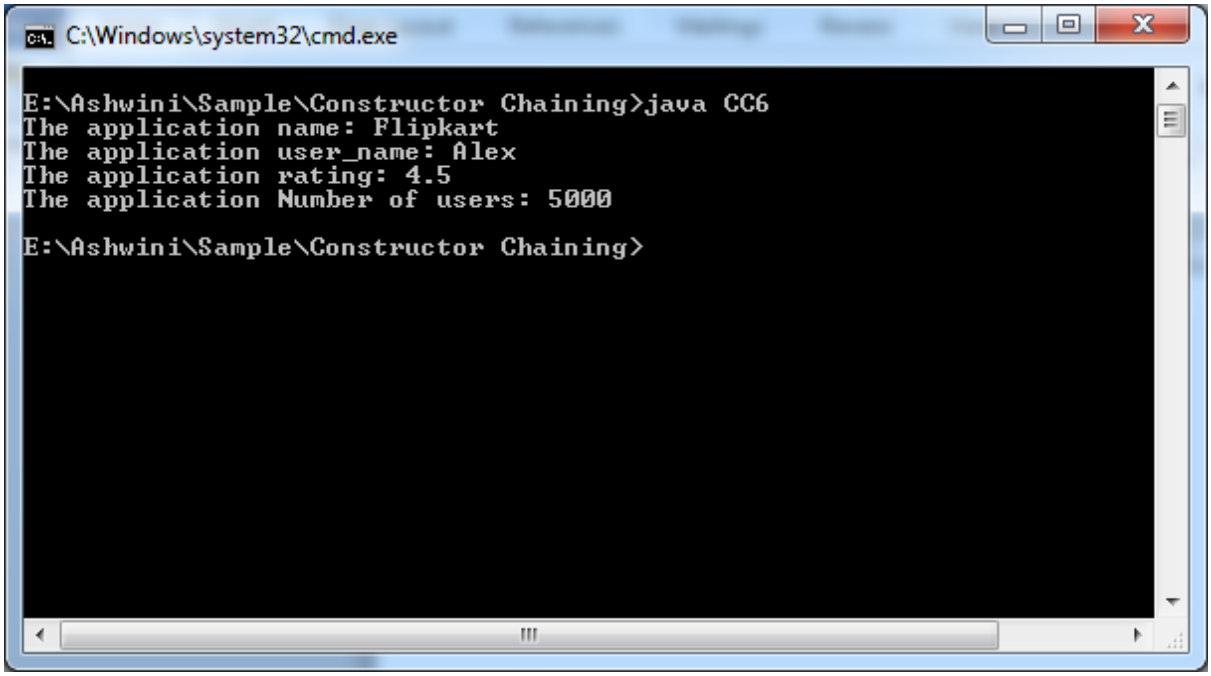
6. Constructor calling within another constructor

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

class B extends A
{
	B()
	{
		System.out.println("The class B constructor ");
	}
}
class C extends B
{
	C()
	{
		System.out.println("The class C constructor ");
	}
}
class CC5
{
	public static void main(String[] args)
	{
		C c=new C();
	}
}
```



7. Write a program to display use of super class constructor

```
class Application
{
	String app_name;
	Application(String app_name)
	{
		this.app_name=app_name;
	}
}

class flipkart extends Application
{
	String Uname;
	double rating;
	int No_of_users;
	flipkart(String Uname,double rating,int No_of_users)
	{
		super("Flipkart");
		this.Uname=Uname;
		this.rating=rating;
		this.No_of_users=No_of_users;
	}
}
class CC6
{
	public static void main(String[] args)
	{
		flipkart f=new flipkart("Alex",4.5,5000);
		System.out.println("The application name: "+f.app_name);
		System.out.println("The application user_name: "+f.Uname);
		System.out.println("The application rating: "+f.rating);
		System.out.println("The application Number of users: "+f.No_of_users);
	}
}
```



6. Program to initialize all 3 constructor without creation of multiple object

```
class Person
{
        String name;
        Person(String name)
        {
                this.name=name;
        }
}

class Employee extends Person
{
        int id;
        String email;
        double salary;

        Employee(String email, double salary)
        {
                this(12);
                this.email=email;
                this.salary=salary;
        }
        Employee(int id)
        {
                super("ALex");
                this.id=id;
        }
}

class CC7
{
        public static void main(String[] args)
        {
                Employee e=new Employee("Ash@gmail.com",5000.00);
```

```
            System.out.println(" The employee id is: " +e.id);
            System.out.println(" The employee email is: " +e.email);
            System.out.println(" The employee salary is: " +e.salary);
            System.out.println(" The employee name is: " +e.name);

        }
}
```