There are two types of constructors in Java:

1. no-arg constructor

2. parameterized constructor.

We don’t need copy contructor in java Bz we can pass object through method as reference.

1. You cannot call a constructor from a method. The only place from which you can invoke constructors using “this()” or, “super()” is the first line of another constructor.

2. But you can call method from contructor.

**Program: 1**

**class** Parent

{

**void** print()

{

System.***out***.println("parent>> i ValueOne=");

}

Parent()

{

System.***out***.println("parent>> i ValueTwo=");

print();

}

}

**public** **class** MyClass **extends** Parent{

**public** **static** **void** main(String... args) {

Parent p1=**new** Parent();

p1.print();

}

}

Output:

parent>> i ValueTwo=

parent>> i ValueOne=

parent>> i ValueOne=

3. You can call method from contructor but see output when we use inheritance

**Program: 2**

**class** Parent

{

**void** print()

{

System.***out***.println("parent>> i ValueOne=");

}

Parent()

{

System.***out***.println("parent>> i ValueTwo=");

print();

}

}

**class** Child **extends** Parent

{

**int** i = 45;

**void** print()

{

System.***out***.println("Child>>i Value = "+i);

}

}

**public** **class** MyClass {

**public** **static** **void** main(String[] args) {

Parent p1=**new** Parent();

p1.print();

Child c1=**new** Child();

c1.print();

}

}

parent>> i ValueTwo=

parent>> i ValueOne=

parent>> i ValueOne=

parent>> i ValueTwo=

Child>>i Value = 0

Child>>i Value = 45

4. You can write return statement inside contructor.

**class** Parent

{

Parent()

{

System.***out***.println("parent>> i ValueTwo=");

**return ;// must write last ,can not at starting**

}

}

**public** **class** MyClass {

**public** **static** **void** main(String[] args) {

Parent p1=**new** Parent();

}

}

4. In Java, if we write our own copy **constructor or parameterized** constructor, then compiler doesn’t create the default constructor. This behavior is same as C++. But we can write only default OR parameterized constructor and defined it.

**class** Parent

{ **int** x;

Parent()

{

System.***out***.println("Default parent Contructor");

**return** ;

}

Parent(Parent p)

{

x=p.x;

System.***out***.println(x);

}

}

**public** **class** MyClass {

**public** **static** **void** main(String[] args) {

Parent p1=**new** Parent**();//we need to defined it Otherwise C.T**

Parent p2=**new** Parent(p1);

}

}

Output:

Default parent Contructor

0

5. **this()** or **super** keyword must be first statement in contructor.

**class** Parent

{ **int** x=10;

Parent(**int** x)

{

x=x;

System.***out***.println(x);

}

}

**public** **class** MyClass {

**public** **static** **void** main(String[] args) {

Parent p2=**new** Parent(20);

}

}Output:20

Java Copy Constructor

There is no copy constructor in Java. However, we can copy the values from one object to another like copy constructor in C++.

There are many ways to copy the values of one object into another in Java. They are:

* By constructor
* By assigning the values of one object into another
* By clone() method of Object class

**By Contructor**

Student6 s1 = **new** Student6(111,"Karan");

Student6 s2 = **new** Student6(s1);

**Without Contructor:**

1. Student7 s1 = **new** Student7(111,"Karan");
2. Student7 s2 = **new** Student7();
3. s2.id=s1.id;
4. s2.name=s1.name;