# **Methods**

**How primitive variables passed to methods - by value or by reference?**

In Java, primitive variables are passed to methods by value. If the passed value changes in the method, it does not change the original value.

Example:

**package** intquestions;

**public** **class** testclass {

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

{

**int** x = 5;

*change*(x);

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

}

**public** **static** **void** change(**int** x)

{

x = 10;

}

}

Result: 5

**When we pass an object to a method in java, does it pass by value or reference?**

Java creates a copy of references and pass it to method, but they still point to same memory reference. Mean if set some other object to reference passed inside method, the object from calling method as well its reference will remain unaffected.

The changes are not reflected back if we change the object itself to refer some other location or object

**package** intquestions;

//A Java program to show that references are also passed

//by value.

**class** Test

{

**int** x;

Test(**int** i) { x = i; }

Test() { x = 0; }

}

**class** testclass

{

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

{

// t is a reference

Test t = **new** Test(5);

// Reference is passed and a copy of reference

// is created in change()

*change*(t);

// Old value of t.x is printed

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

}

**public** **static** **void** change(Test t)

{

// We changed reference to refer some other location

// now any changes made to reference are not reflected

// back in main

t = **new** Test();

t.x = 10;

}

}