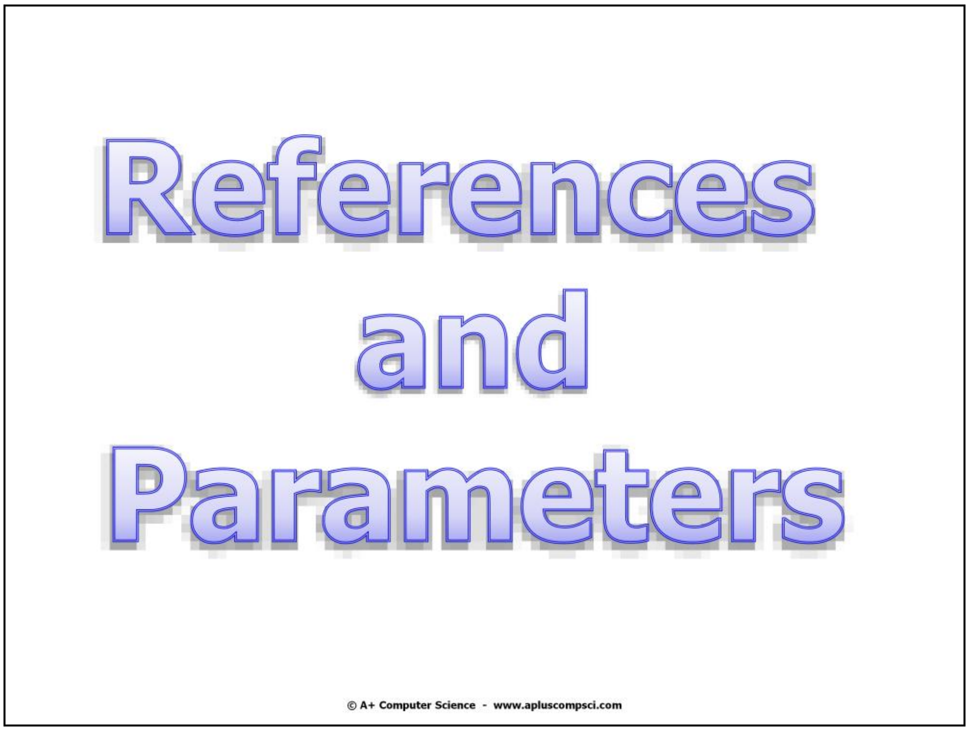
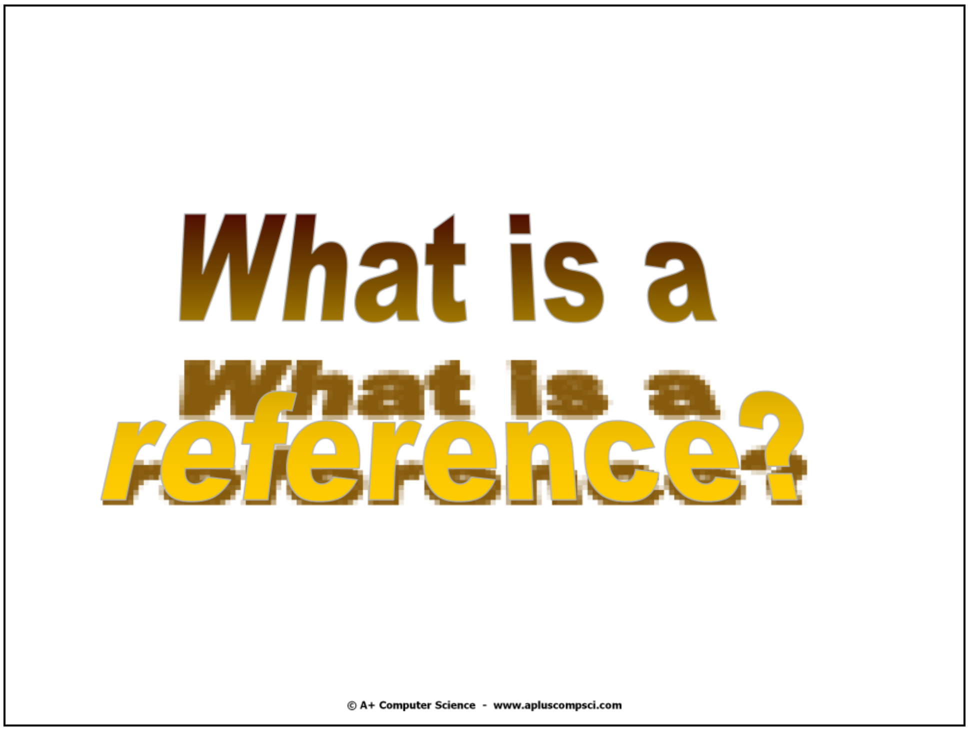
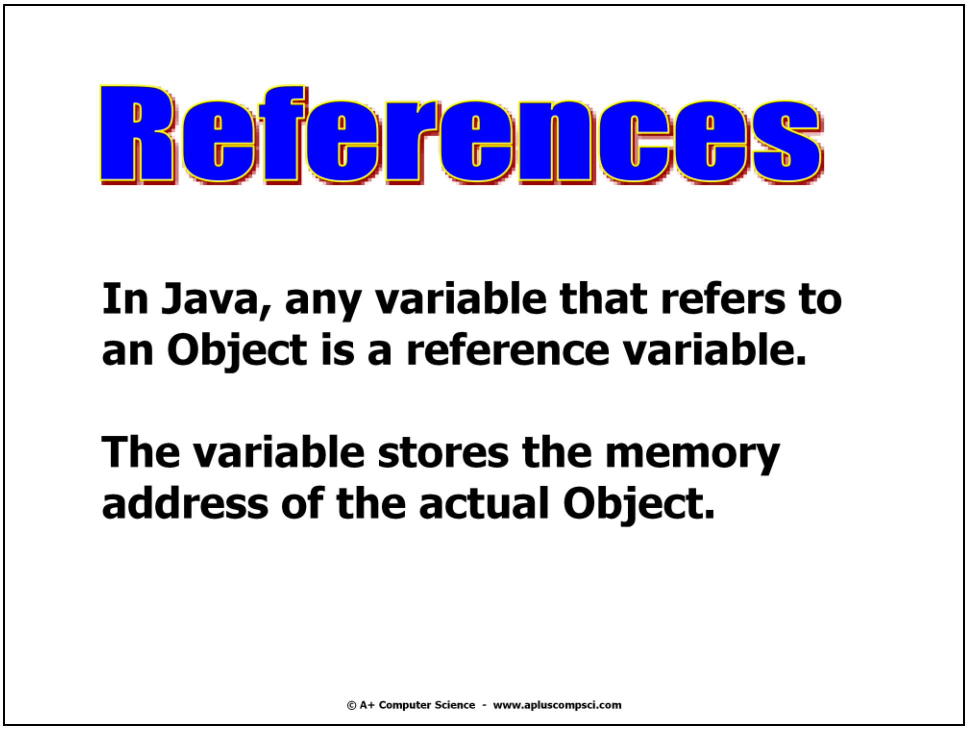


# References and Parameters



## What is a reference?



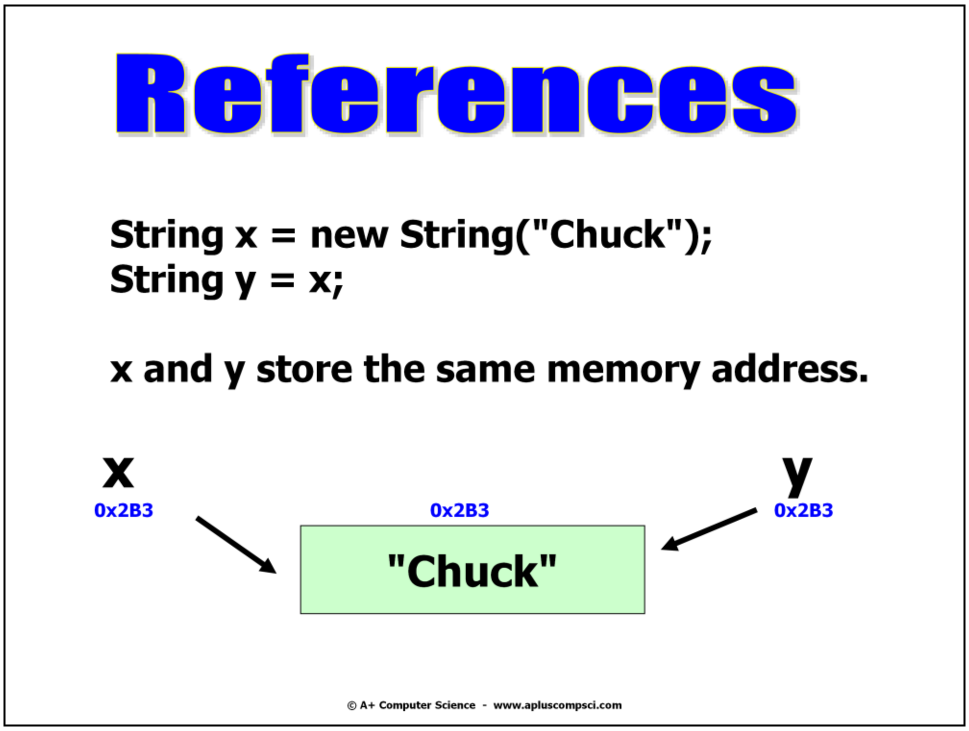
## References

**In Java, any variable that refers to an Object is a reference variable.**

**The variable stores the memory address of the actual Object.**

© A+ Computer Science - www.apluscompsci.com

All variables in Java that refer to Objects are called references. Reference variables store the location / memory address of the actual Object. For most situations, the value stored in a reference is a memory address.



## References

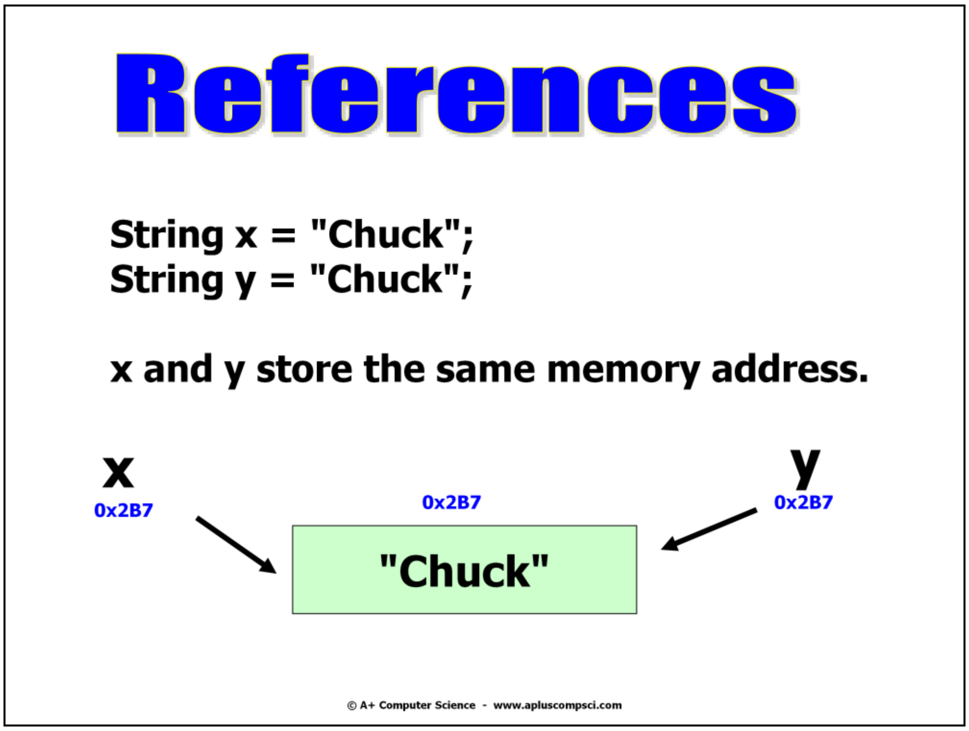
```
String x = new String("Chuck");
String y = x;
```

x and y store the same memory address.

In this example, x and y both the store the location / address of `Chuck`. There is only one String containing `Chuck`. There are two reference variables storing the location / address of `Chuck`.

For this example, `x==y` is true. `x==y` compares the values stored in x and y. x and y both store the same location / address.

For this example, `x.equals(y)` is true. `x.equals(y)` compares the contents of the Objects referred to by x and y. `Chuck` is being compare to `Chuck`.



## References

```
String x = "Chuck";
String y = "Chuck";
```

x and y store the same memory address.

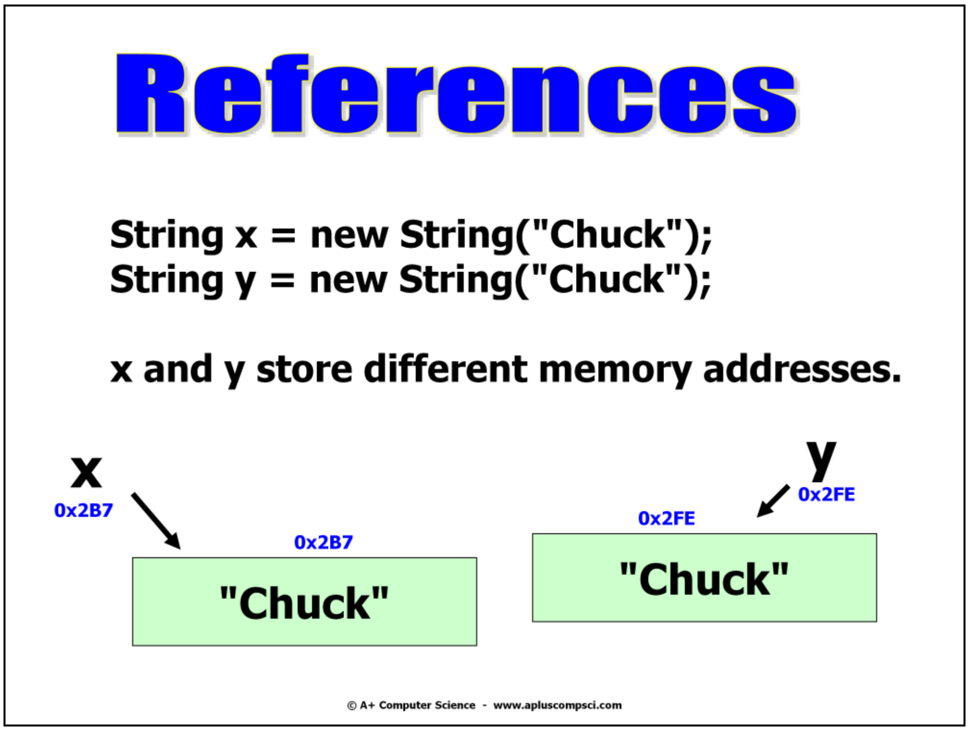
x
0x2B7
0x2B7
"Chuck"
y
0x2B7

© A+ Computer Science - www.apluscompsci.com

In this example, x and y both the store the location of `Chuck`. There is only one String containing `Chuck`. There are two reference variables storing the location / address of `Chuck`.

For this example, `x==y` is true. `x==y` compares the values stored in x and y. x and y both store the same location / address.

For this example, `x.equals(y)` is true. `x.equals(y)` compares the contents of the Objects referred to by x and y. `Chuck` is being compare to `Chuck`.



## References

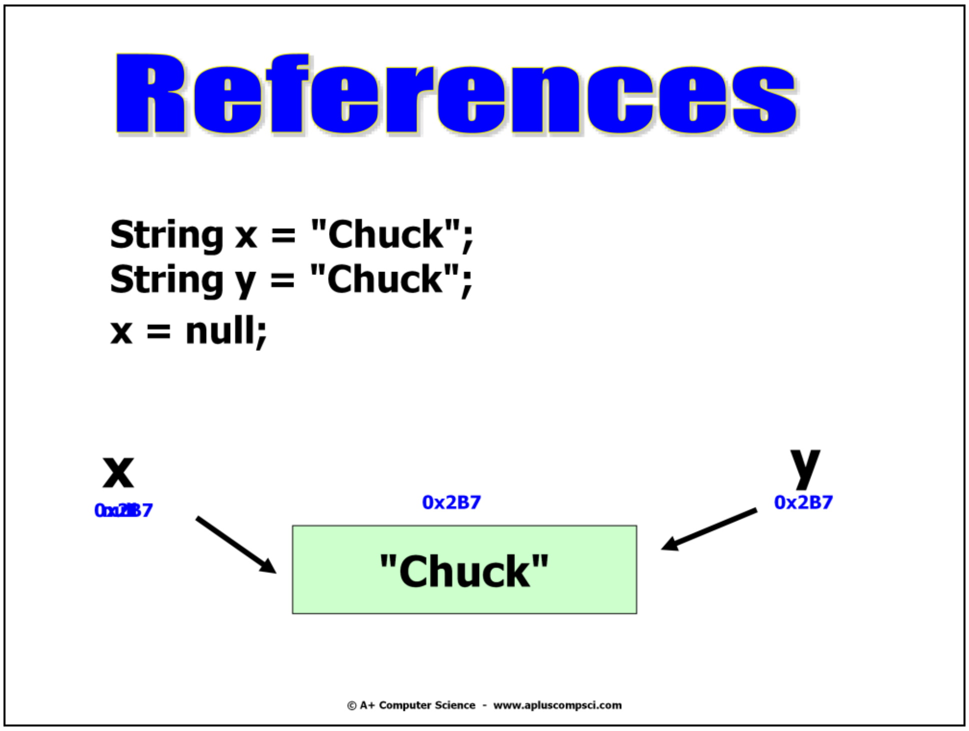
```
String x = new String("Chuck");
String y = new String("Chuck");
```

x and y store different memory addresses.

In this example, x stores the location / address of a String Object that stores the value `Chuck`. y also stores the location of a different String Object that stores the value `Chuck`. x and y do not store the same location / address.

For this example, `x==y` is false. x and y do not store the same location / address.

For this example, `x.equals(y)` is true.



## References

```
String x = "Chuck";
String y = "Chuck";
x = null;
```

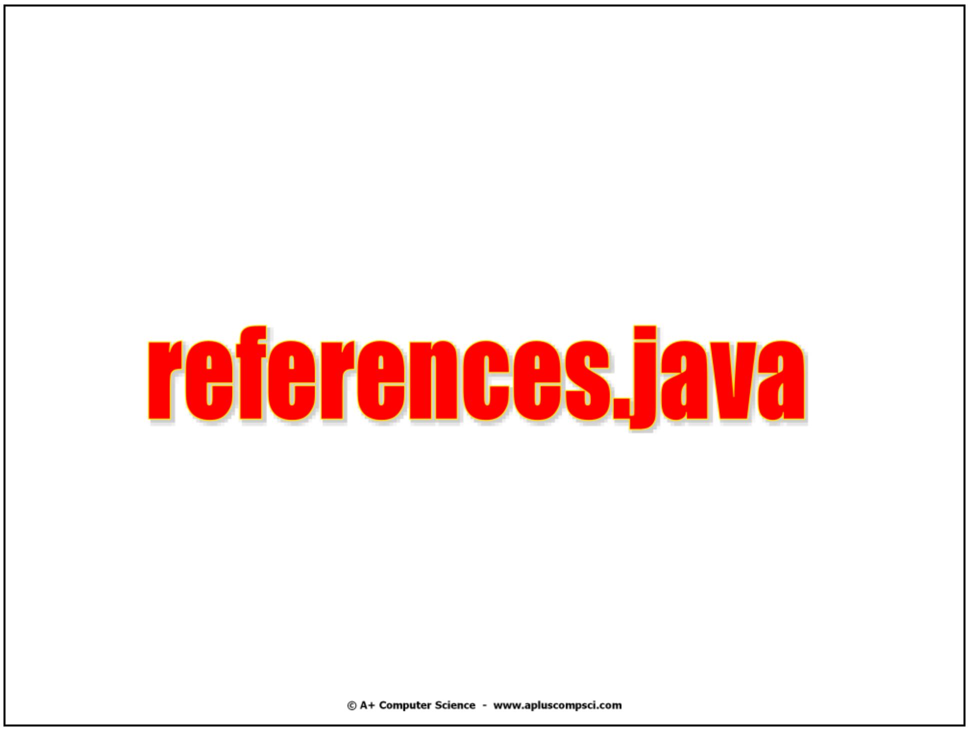
x
0x2B7
0x2B7
"Chuck"
y
0x2B7

In this example, x and y both the store the location / address of `Chuck`. There is only one String containing `Chuck`. There are two reference variables storing the location / address of `Chuck`.

At the start, `x==y` is true.

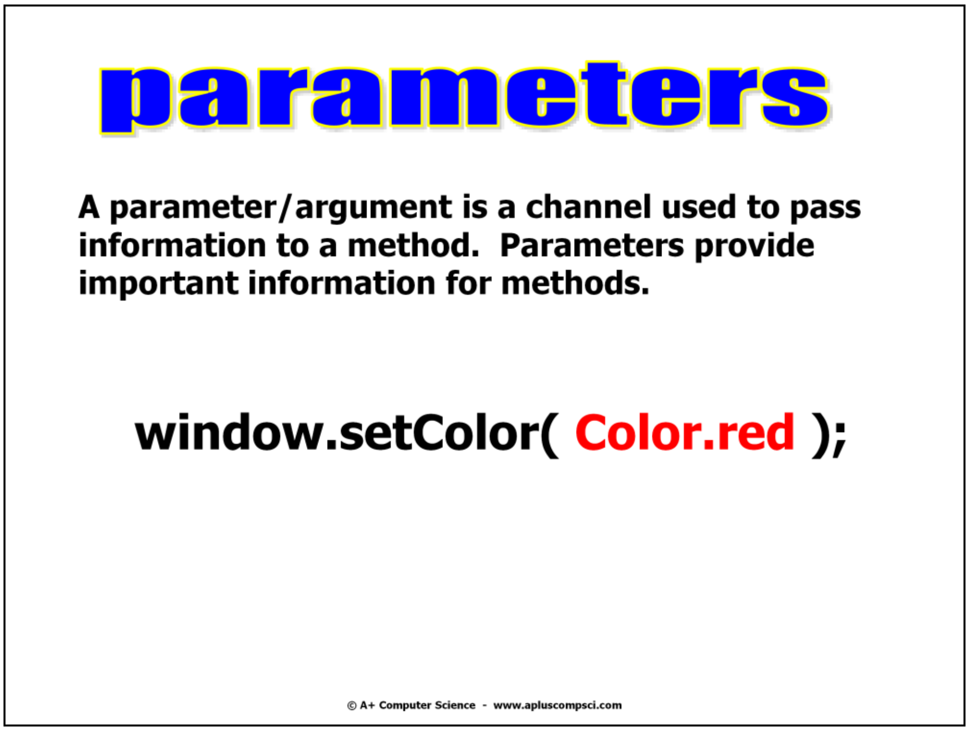
x is then referred to null. x now stores null. y was in no way changed. y still stores the address of `Chuck`.

After changing the value of x, `x==y` is false.



## references.java

## What is a parameter?

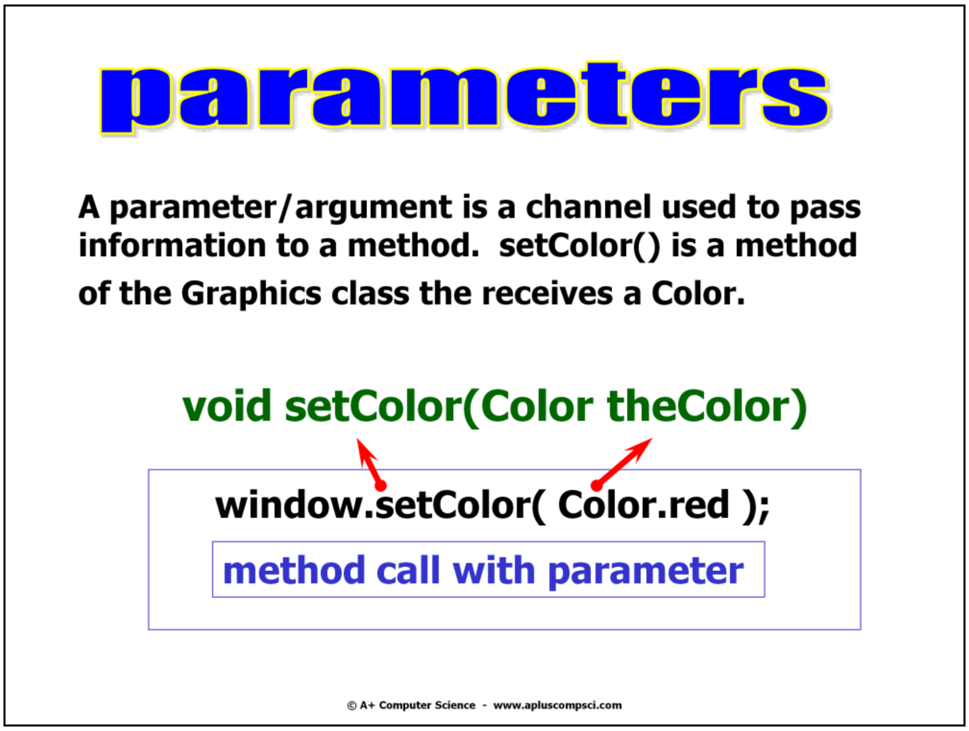


## parameters

**A parameter/argument is a channel used to pass information to a method. Parameters provide important information for methods.**

```
window.setColor( Color.red );
```

Parameters are used to pass information to a method. Many methods need information in order to perform some operation. The parameters tell the method what to do and how to do it.



## parameters

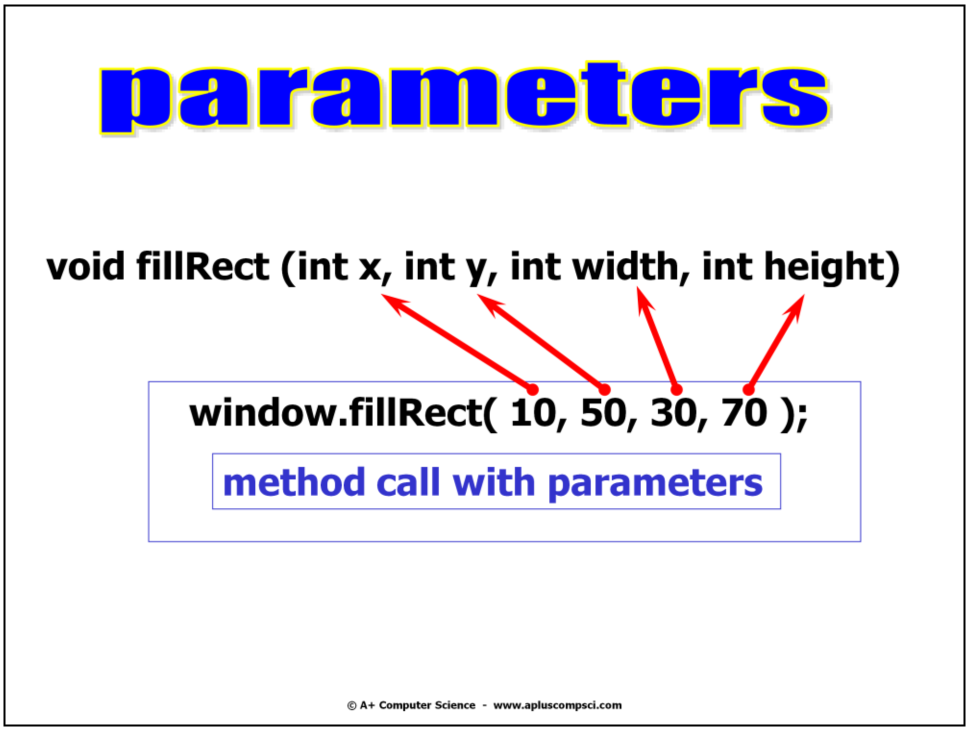
A parameter/argument is a channel used to pass information to a method. setColor() is a method of the Graphics class the receives a Color.

```
void setColor(Color theColor)

window.setColor( Color.red );
```

method call with parameter

Most, if not all, of the `Graphics` class methods require parameters. The parameters communicate to the `Graphics` methods information about what needs to be done. The `setColor()` method changes the current drawing color to the color passed in. `setColor()` cannot be called without a color parameter.



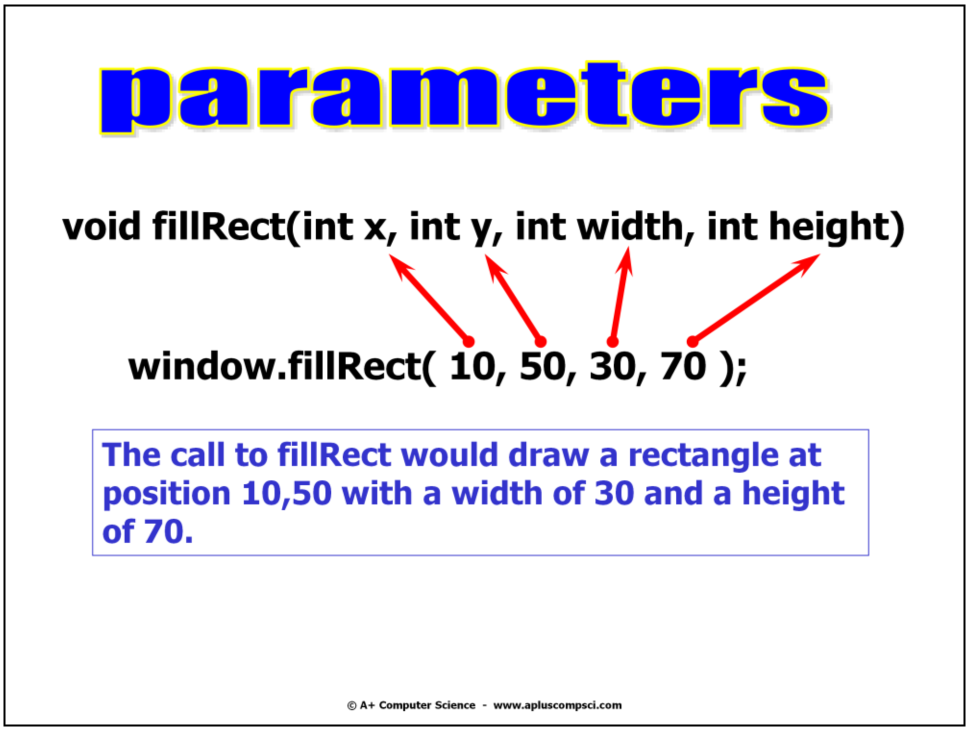
## parameters

```
void fillRect (int x, int y, int width, int height)

window.fillRect( 10, 50, 30, 70 );
```

method call with parameters

The `fillRect()` method requires four pieces of information. `fillRect()` needs an x value, a y value, a width, and a heigth. `fillRect()` will draw a filled rectangle on the window at x,y with height and width as stated by the parameters.

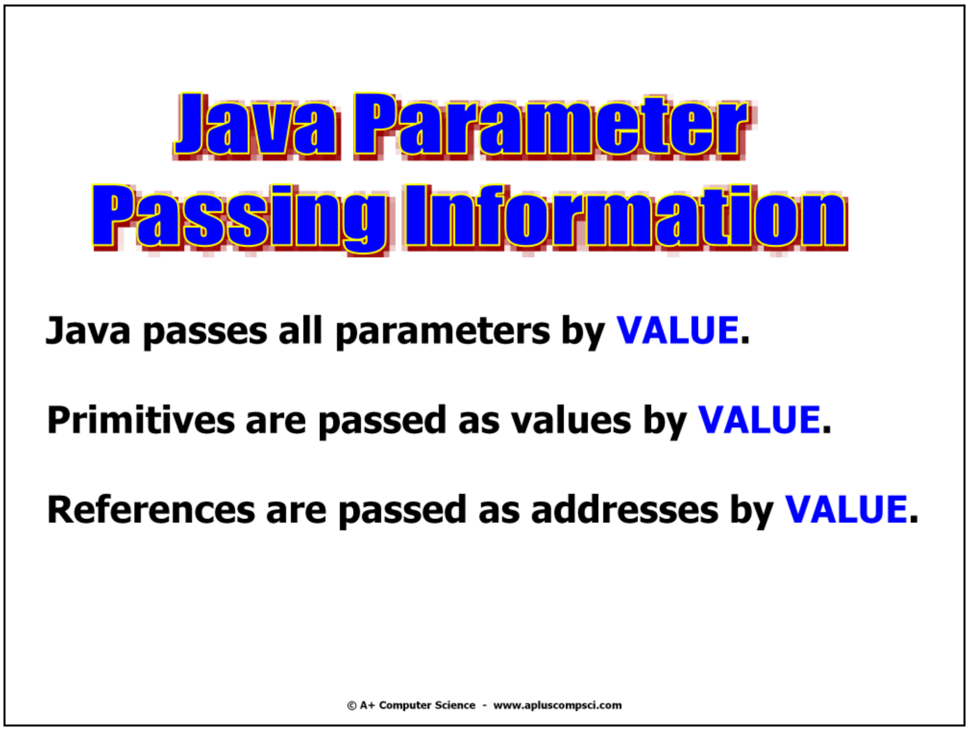


## parameters

```
void fillRect(int x, int y, int width, int height)

window.fillRect( 10, 50, 30, 70 );
```

The call to fillRect would draw a rectangle at position 10,50 with a width of 30 and a height of 70.



## Java Parameter Passing Information

**Java passes all parameters by VALUE.**

**Primitives are passed as values by VALUE.**

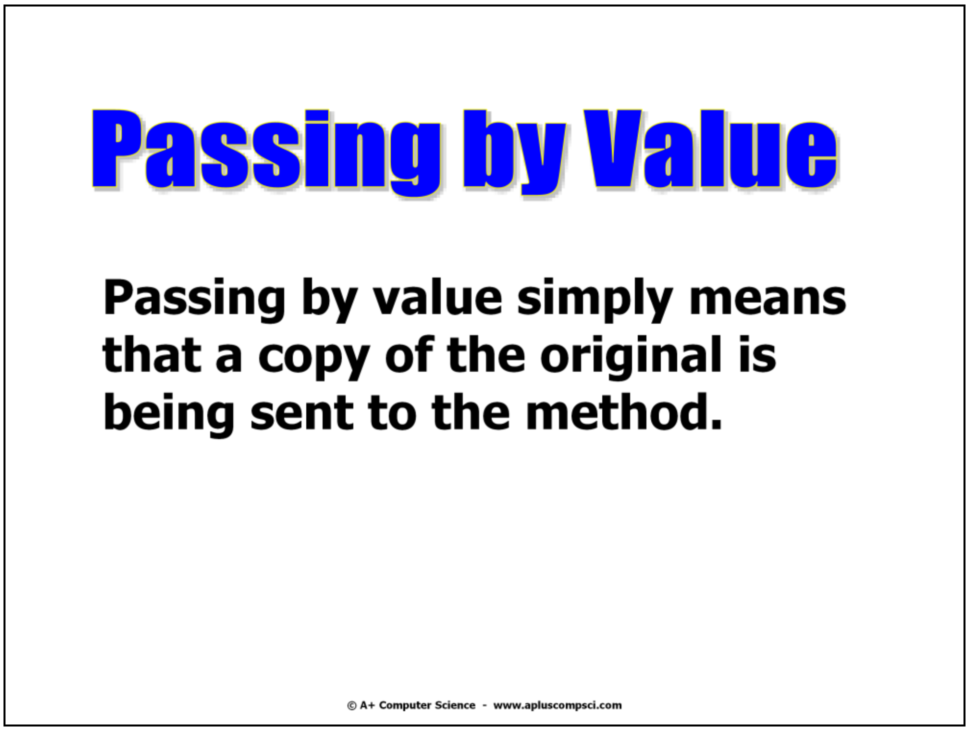
**References are passed as addresses by VALUE.**

© A+ Computer Science - www.apluscompsci.com

Java passes all parameters by VALUE.

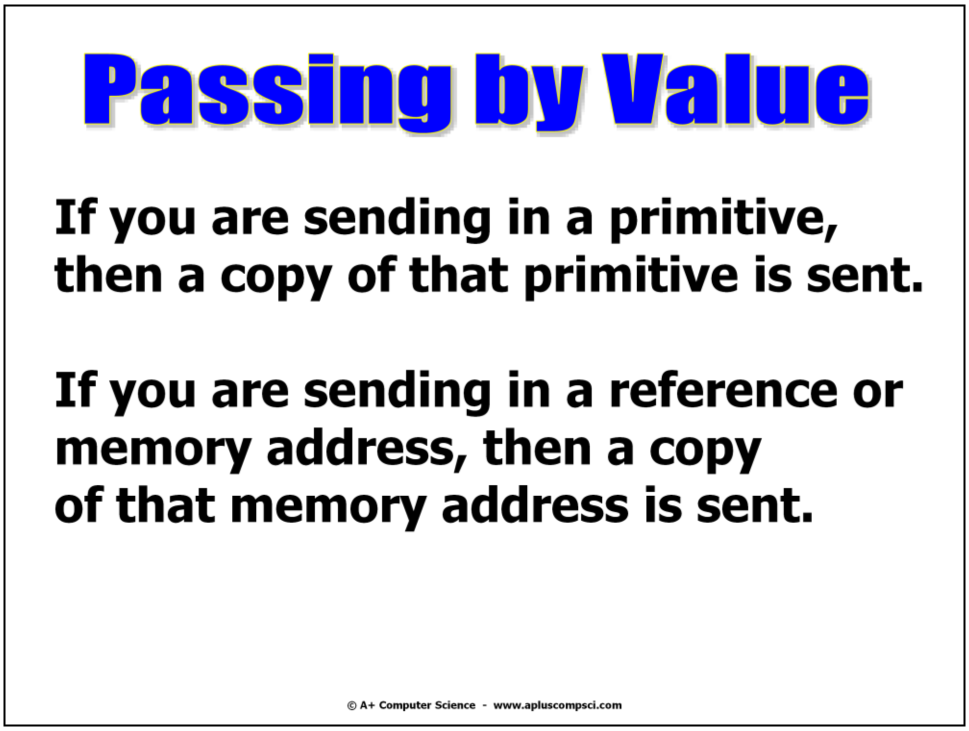
What does that mean?

Java makes a copy of what is being passed to the method. Because a copy of the original value is sent in, the original value sent in cannot be changed.



## Passing by Value

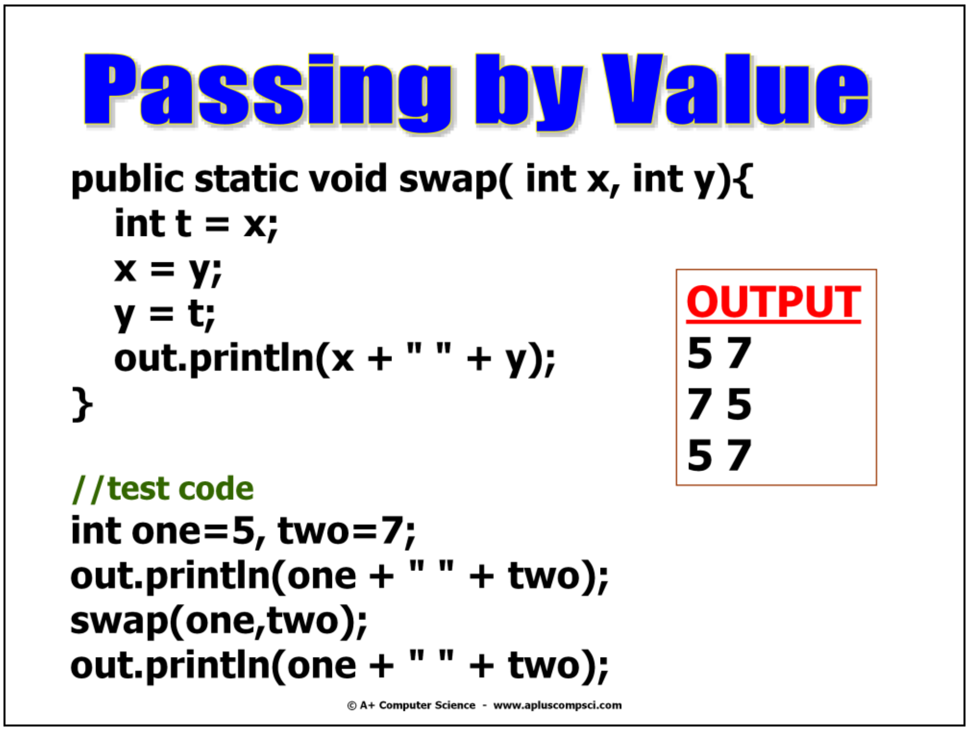
Passing by value simply means that a copy of the original is being sent to the method.



## Passing by Value

If you are sending in a primitive, then a copy of that primitive is sent.

If you are sending in a reference or memory address, then a copy of that memory address is sent.



## Passing by Value

```
public static void swap( int x, int y){
   int t = x;
   x = y;
   y = t;
   out.println(x + " " + y);
}

//test code
int one=5, two=7;
out.println(one + " " + two);
swap(one,two);
out.println(one + " " + two);
```

OUTPUT

```
5 7
7 5
5 7
```

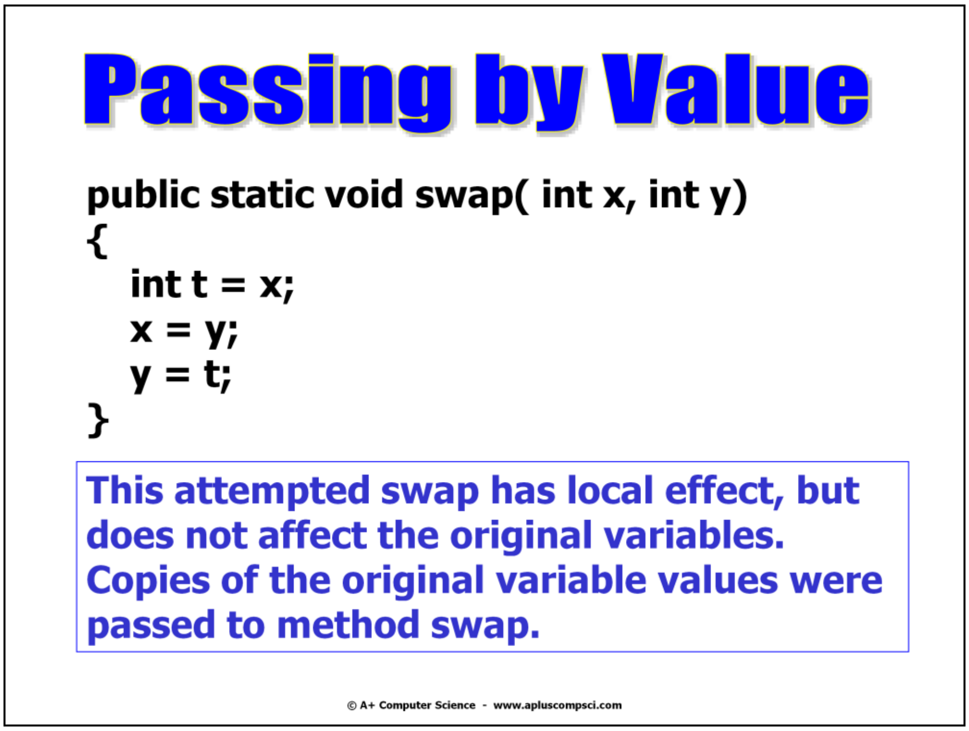
© A+ Computer Science - www.apluscompsci.com

The original value of one is copied and pasted into x. The original value of two is copied and pasted into y.

What is the value of one? One stores a simple integer value.

There is no connection between one and x other than the fact that they both store the same value. The same is true for two and y.

Changes made to x do not have any affect on one. Changes made to y do not have any affect on two.



## Passing by Value

```
public static void swap( int x, int y)
{
   int t = x;
   x = y;
   y = t;
}
```

This attempted swap has local effect, but does not affect the original variables. Copies of the original variable values were passed to method swap.

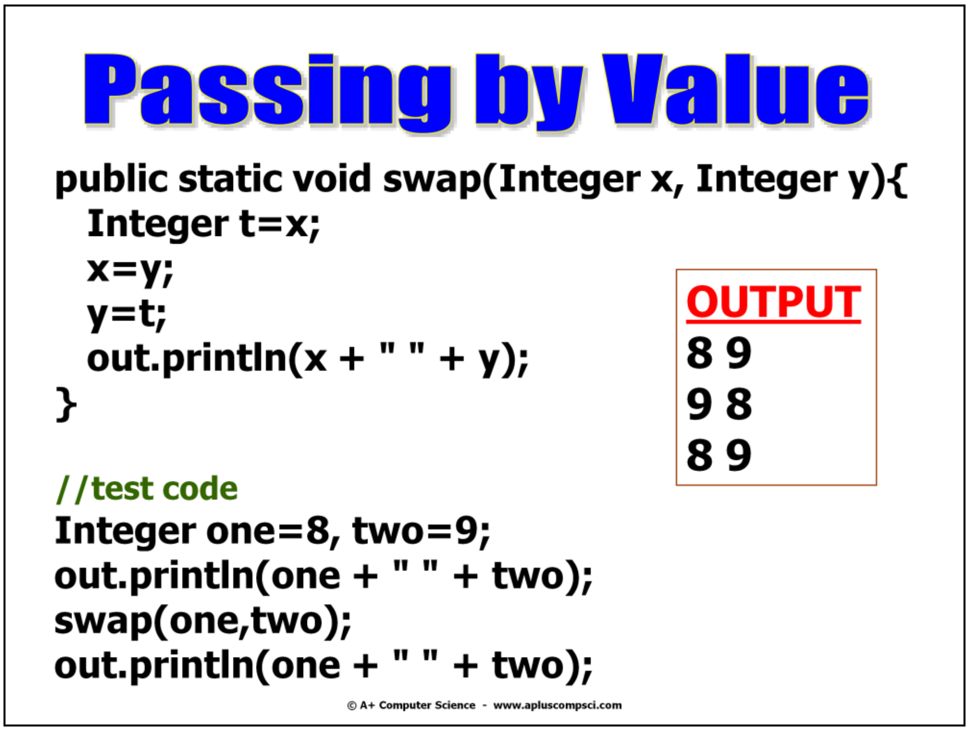
© A+ Computer Science - www.apluscompsci.com

The original value of one is copied and pasted into x. The original value of two is copied and pasted into y.

What is the value of one? One stores a simple integer value.

There is no connection between one and x other than the fact that they both store the same value. The same is true for two and y.

Changes made to x do not have any affect on one. Changes made to y do not have any affect on two.



## Passing by Value

```
public static void swap(Integer x, Integer y){
   Integer t=x;
   x=y;
   y=t;
   out.println(x + " " + y);
}

//test code
Integer one=8, two=9;
out.println(one + " " + two);
swap(one,two);
out.println(one + " " + two);
```

OUTPUT

```
8 9
9 8
8 9
```

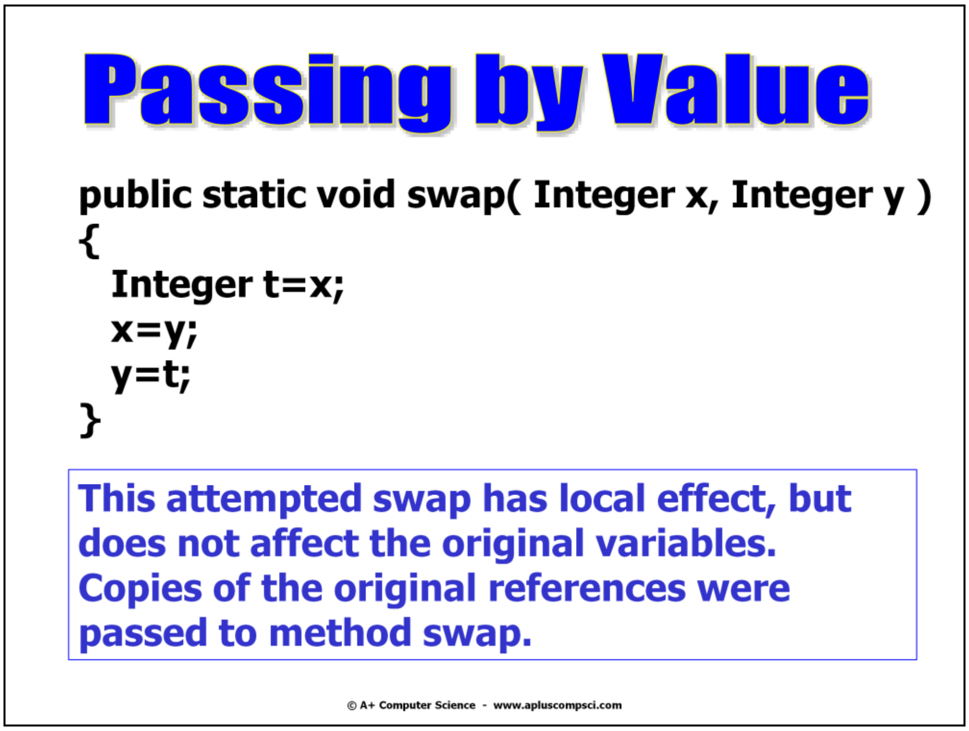
© A+ Computer Science - www.apluscompsci.com

The original value of one is copied and pasted into x. The original value of two is copied and pasted into y.

What is the value of one? One stores the location/address of an Integer Object.

There is no connection between one and x other than the fact that they both store the same location/address. The same is true for two and y.

Changes made to x do not have any affect on one. Changes made to y do not have any affect on two.



## Passing by Value

```
public static void swap( Integer x, Integer y )
{
   Integer t=x;
   x=y;
   y=t;
}
```

This attempted swap has local effect, but does not affect the original variables. Copies of the original references were passed to method swap.

© A+ Computer Science - www.apluscompsci.com

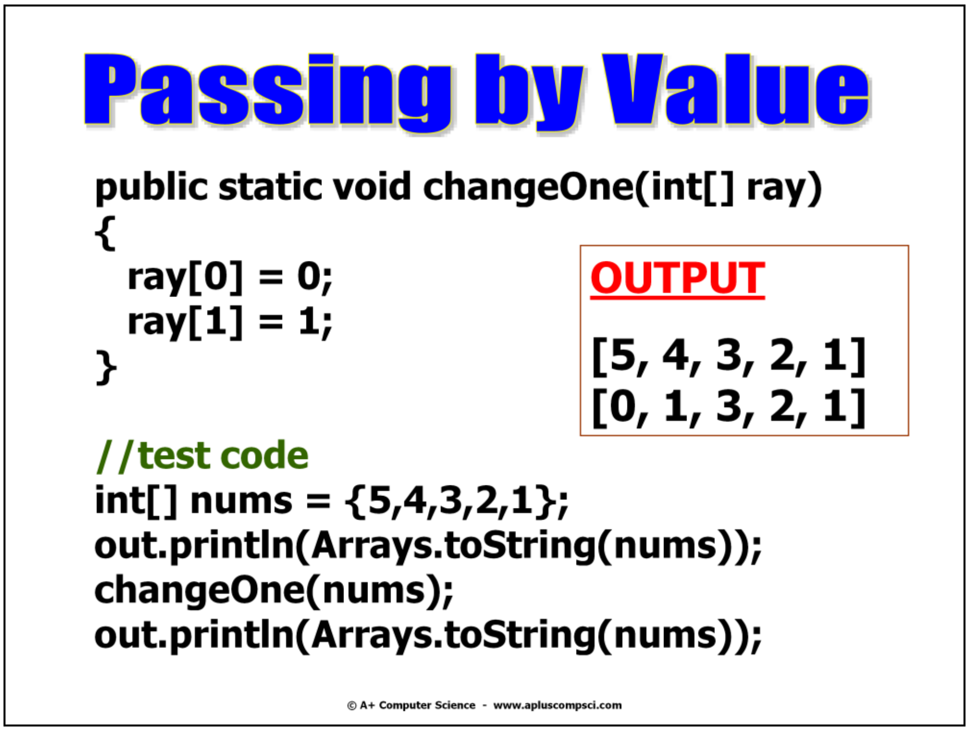
The original value of one is copied and pasted into x. The original value of two is copied and pasted into y.

What is the value of one? One stores the location/address of an Integer Object.

There is no connection between one and x other than the fact that they both store the same location/address. The same is true for two and y.

Changes made to x do not have any affect on one. Changes made to y do not have any affect on two.

## passbyvalueone.java



## Passing by Value

```
public static void changeOne(int[] ray)
{
   ray[0] = 0;
   ray[1] = 1;
}

//test code
int[] nums = {5,4,3,2,1};
out.println(Arrays.toString(nums));
changeOne(nums);
out.println(Arrays.toString(nums));
```

**OUTPUT**

```
[5, 4, 3, 2, 1]
[0, 1, 3, 2, 1]
```

© A+ Computer Science - www.apluscompsci.com

The original value of nums is copied and pasted into ray.

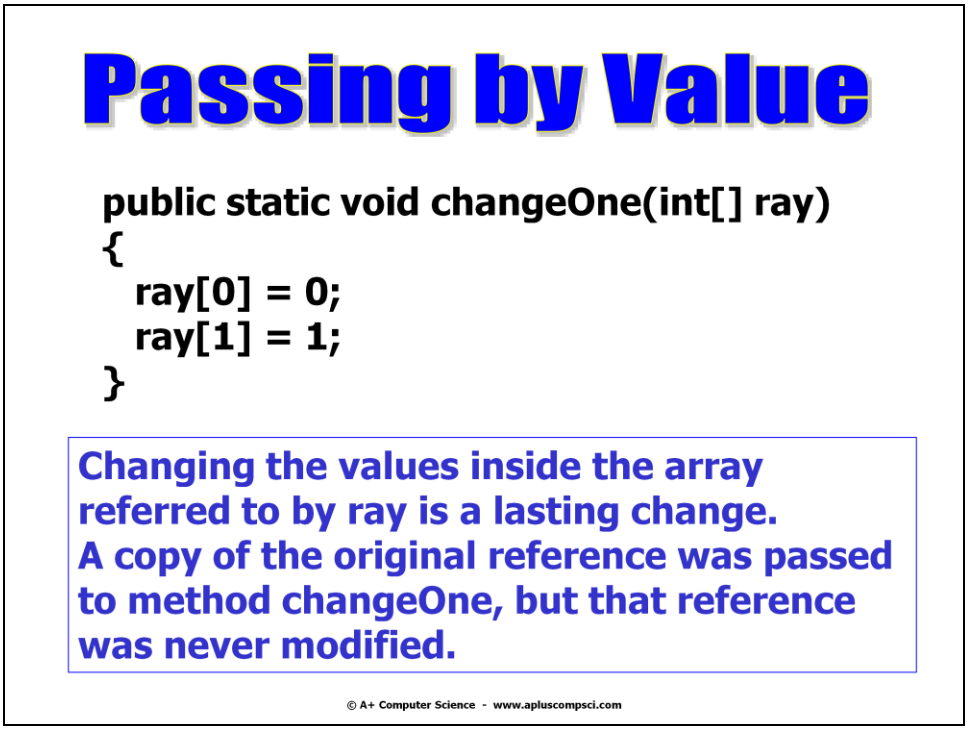
What is the value of nums? Nums stores the location/address of an int[] array.

Nums and ray store the same location/address of the same int[] array.

Changing the location/address of ray would in no way affect the location/address stored in nums.

Changing the values in the int[] array that both refer to does affect both.

Both nums and ray store the exact same location/address of the exact same int[] array.



## Passing by Value

```
public static void changeOne(int[] ray)
{
  ray[0] = 0;
  ray[1] = 1;
}
```

**Changing the values inside the array referred to by ray is a lasting change.
A copy of the original reference was passed to method changeOne, but that reference was never modified.**

The original value of nums is copied and pasted into ray.

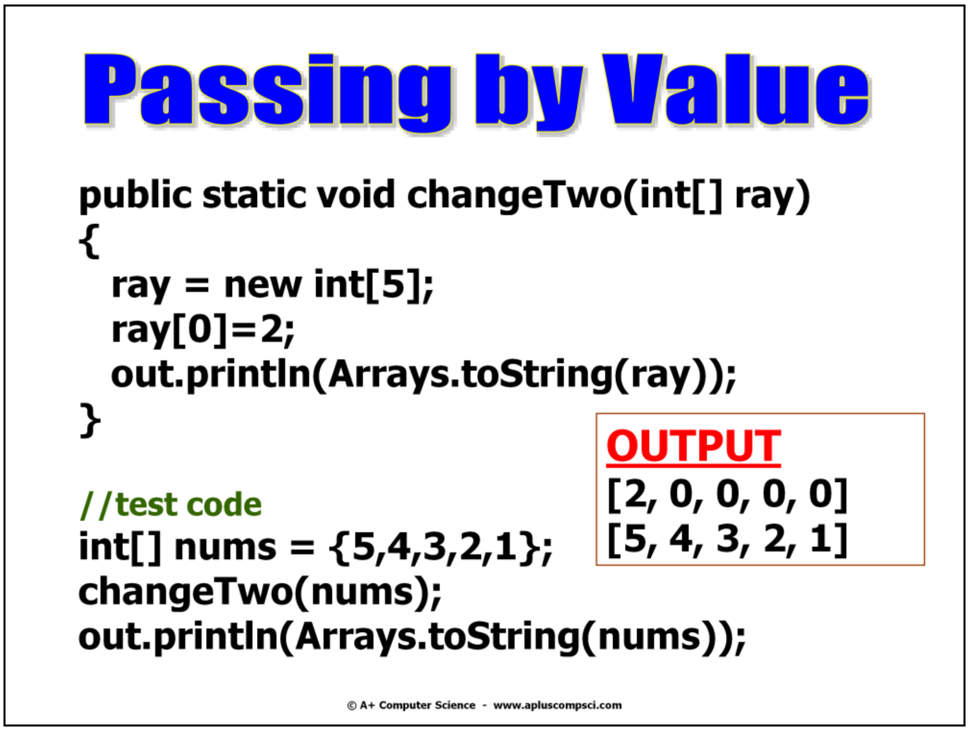
What is the value of nums? Nums stores the location/address of an int[] array.

Nums and ray store the same location/address of the same int[] array.

Changing the location/address of ray would in no way affect the location/address stored in nums.

Changing the values in the int[] array that both refer to does affect both.

Both nums and ray store the exact same location/address of the exact same int[] array.



## Passing by Value

```
public static void changeTwo(int[] ray)
{
  ray = new int[5];
  ray[0]=2;
  out.println(Arrays.toString(ray));
}

//test code
int[] nums = {5,4,3,2,1};
changeTwo(nums);
out.println(Arrays.toString(nums));
```

OUTPUT

```
[2, 0, 0, 0, 0]
[5, 4, 3, 2, 1]
```

© A+ Computer Science - www.apluscompsci.com

The original value of nums is copied and pasted into ray.

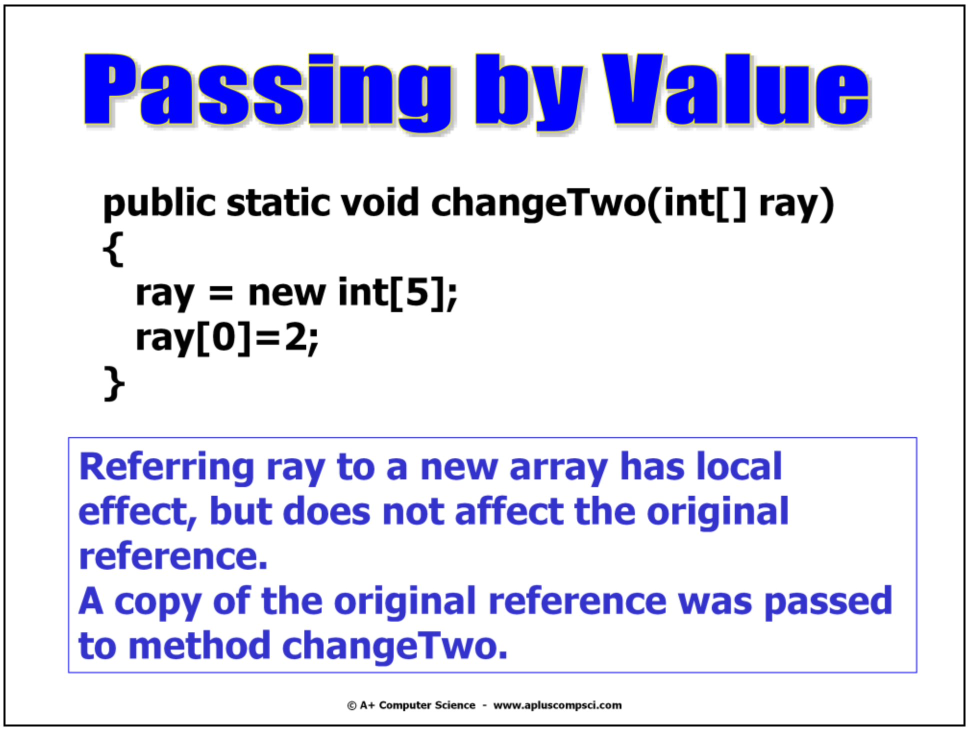
What is the value of nums? Nums stores the location/address of an int[] array.

Nums and ray store the same location/address of the same int[] array.

Changing the location/address of ray does not affect the location/address stored in nums.

The code in changeTwo refers ray to a new int[] array. This change to ray does not affect nums as nums' value was copied and pasted into ray.

All changes made to the new array that ray refers to have affect in method changeTwo only.



## Passing by Value

```
public static void changeTwo(int[] ray)
{
  ray = new int[5];
  ray[0]=2;
}
```

Referring ray to a new array has local effect, but does not affect the original reference.
A copy of the original reference was passed to method changeTwo.

© A+ Computer Science - www.apluscompsci.com

The original value of nums is copied and pasted into ray.

What is the value of nums? Nums stores the location/address of an int[] array.

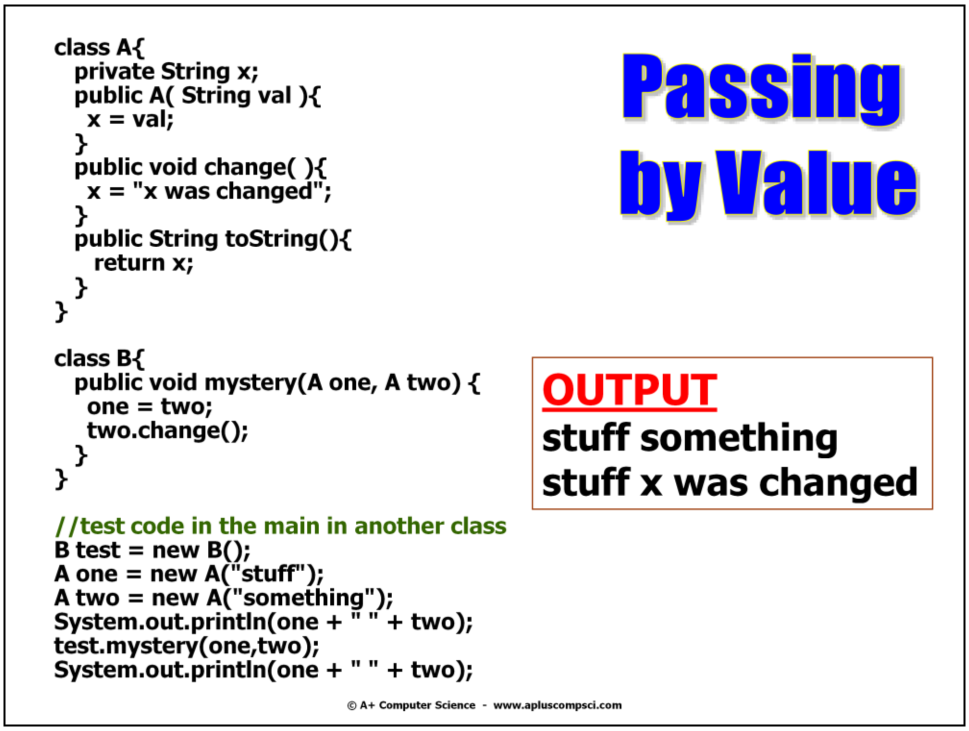
Nums and ray store the same location/address of the same int[] array.

Changing the location/address of ray does not affect the location/address stored in nums.

The code in changeTwo refers ray to a new int[] array. This change to ray does not affect nums as nums' value was copied and pasted into ray.

All changes made to the new array that ray refers to have affect in method changeTwo only.

passbyvaluetwo.java



## Passing by Value

```
class A{
  private String x;
  public A( String val ){
    x = val;
  }
  public void change( ){
    x = "x was changed";
  }
  public String toString(){
     return x;
  }
}

class B{
  public void mystery(A one, A two) {
    one = two;
    two.change();
  }
}

//test code in the main in another class
B test = new B();
A one = new A("stuff");
A two = new A("something");
System.out.println(one + " " + two);
test.mystery(one,two);
System.out.println(one + " " + two);
```

**OUTPUT**

```
stuff something
stuff x was changed
```

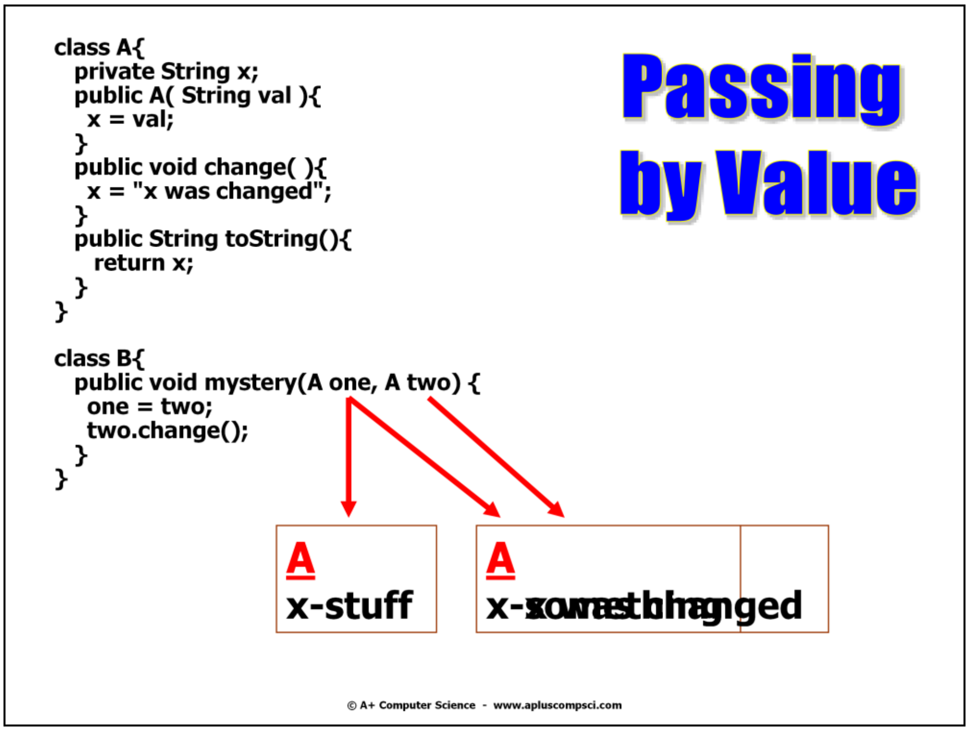
© A+ Computer Science - www.apluscompsci.com

The original values of one and two are copied and pasted into mystery.

What is the value of one? One stores the location/address of an A as does two.

In mystery, one gets the location/address of one from the main and two gets the location/address of two from the main.

In mystery, the location/address of two is copied to one. This change has local affect only. Calling the change method on two changes the instance variable x inside of the A Object referred to by two. This change affects the entire program as it changes the Object referred to and not the actual reference.



## Passing by Value

```
class A{
  private String x;
  public A( String val ){
    x = val;
  }
  public void change( ){
    x = "x was changed";
  }
  public String toString(){
    return x;
  }
}

class B{
  public void mystery(A one, A two) {
    one = two;
    two.change();
  }
}
```

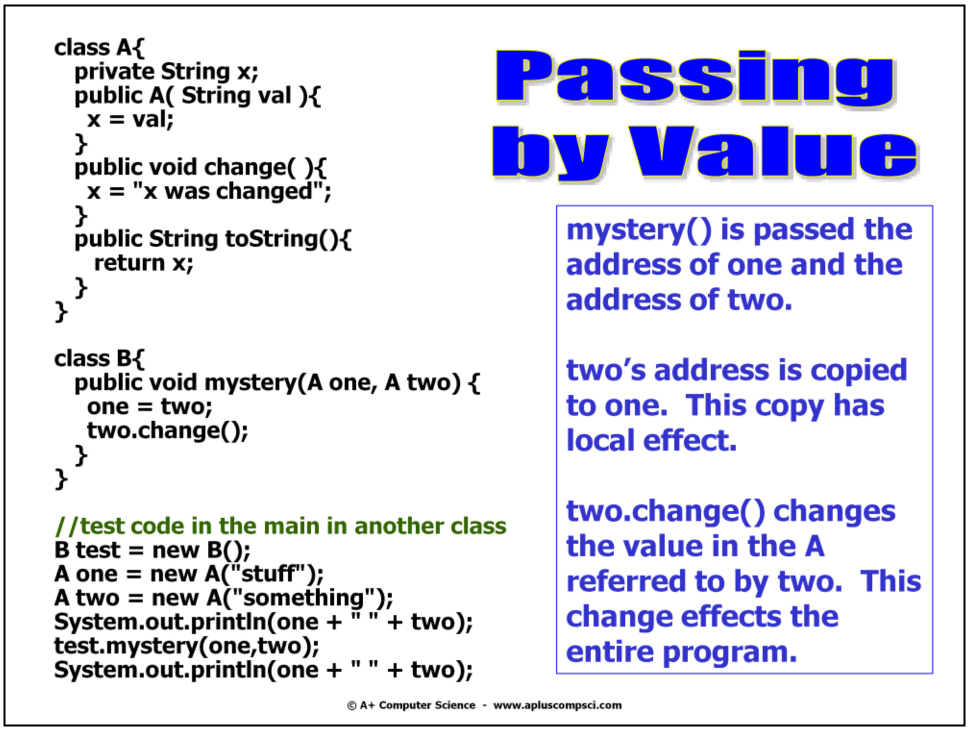
A
x-stuff
A
x-somethingx was changed

The original values of one and two are copied and pasted into mystery.

What is the value of one? One stores the location/address of an A as does two.

In mystery, one gets the location/address of one from the main and two gets the location/address of two from the main.

In mystery, the location/address of two is copied to one. This change has local affect only. Calling the change method on two changes the instance variable x inside of the A Object referred to by two. This change affects the entire program as it changes the Object referred to and not the actual reference.



## Passing by Value

```
class A{
  private String x;
  public A( String val ){
   x = val;
  }
  public void change( ){
   x = "x was changed";
  }
  public String toString(){
    return x;
  }
}

class B{
  public void mystery(A one, A two) {
   one = two;
   two.change();
  }
}

//test code in the main in another class
B test = new B();
A one = new A("stuff");
A two = new A("something");
System.out.println(one + " " + two);
test.mystery(one,two);
System.out.println(one + " " + two);
```

mystery() is passed the address of one and the address of two.

two's address is copied to one. This copy has local effect.

two.change() changes the value in the A referred to by two. This change effects the entire program.

© A+ Computer Science - www.apluscompsci.com

The original values of one and two are copied and pasted into mystery.

What is the value of one? One stores the location/address of an A as does two.

In mystery, one gets the location/address of one from the main and two gets the location/address of two from the main.

In mystery, the location/address of two is copied to one. This change has local affect only. Calling the change method on two changes the instance variable x inside of the A Object referred to by two. This change affects the entire program as it changes the Object referred to and not the actual reference.

## passbyvaluethree.java

© A+ Computer Science - www.apluscompsci.com

## Start work on the labs