

References and Parameters

Lab 17

What is a
What is a
reference?

References

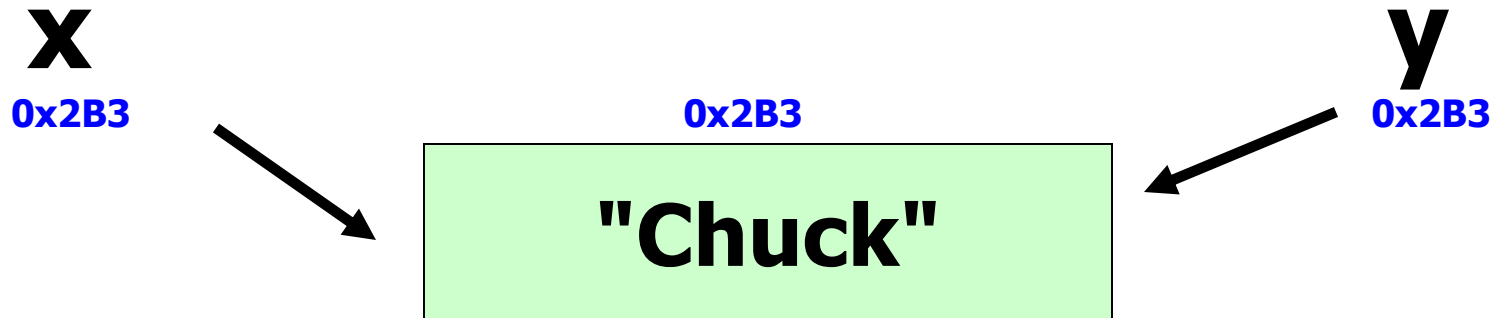
In Java, any variable that refers to an Object is a reference variable.

The variable stores the memory address of the actual Object.

References

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

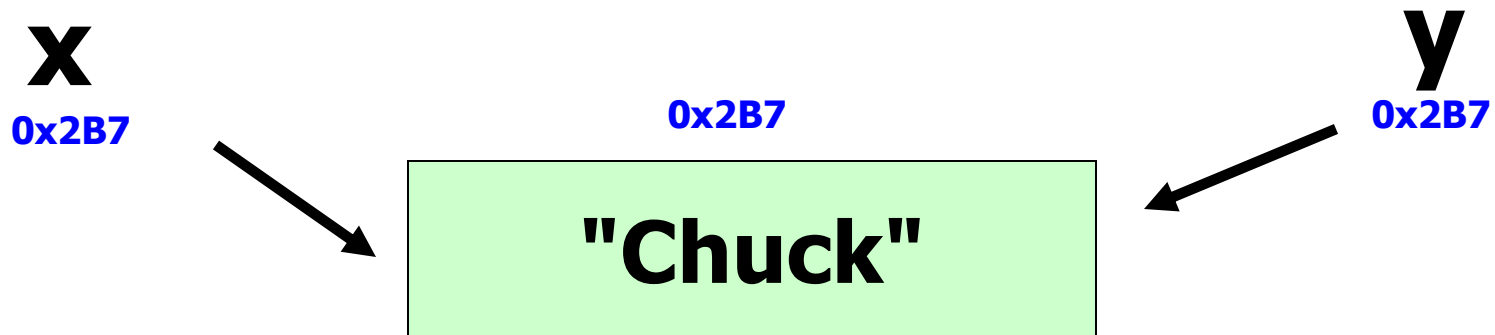
x and y store the same memory address.



References

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

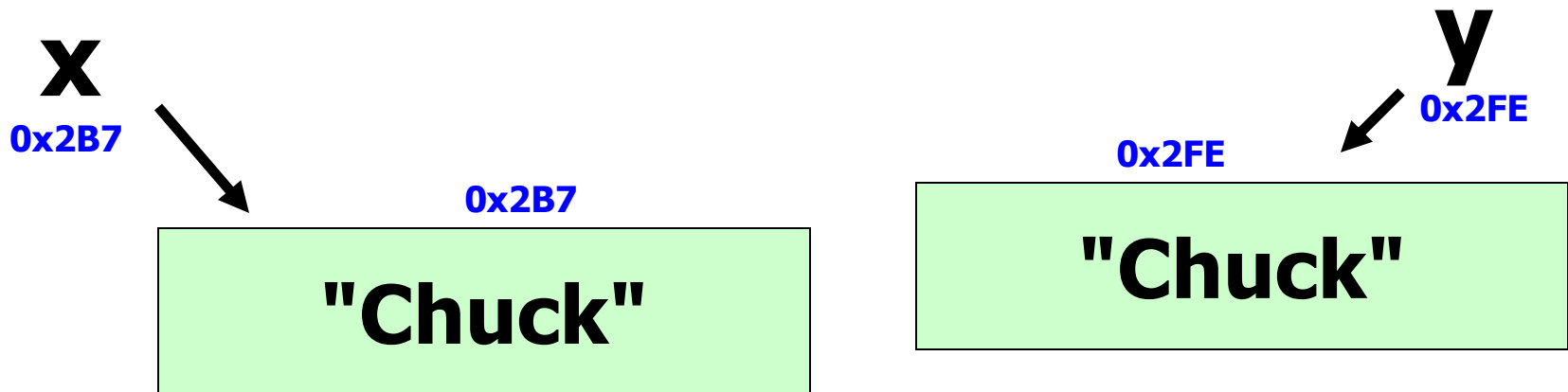
x and y store the same memory address.



References

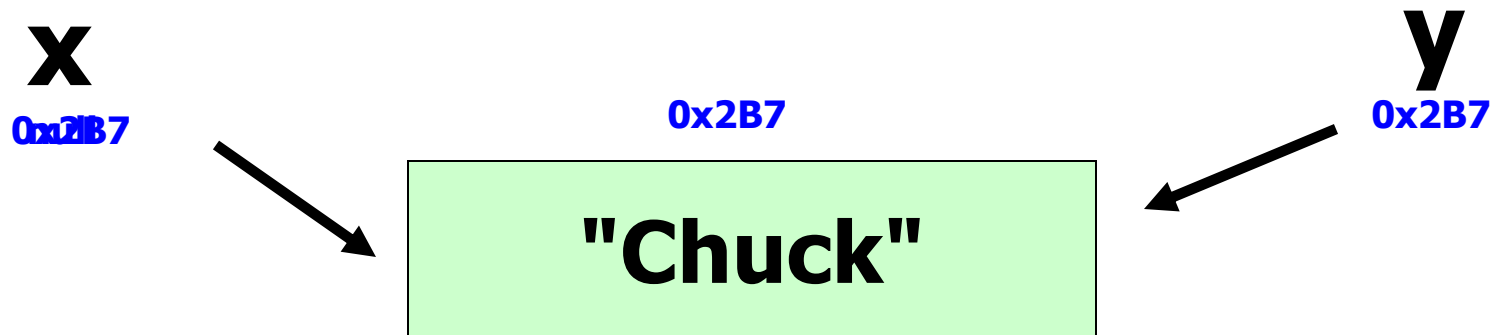
```
String x = new String("Chuck");  
String y = new String("Chuck");
```

x and y store different memory addresses.



References

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



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

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

void setColor(Color theColor)



The diagram illustrates the relationship between a method signature and its call. At the top, the method signature `void setColor(Color theColor)` is shown in green. Below it, a blue-bordered box contains the method call `window.setColor(Color.red);`. Two red arrows point from the arguments in the call to the parameter in the signature: one from `Color` to `Color`, and another from `red` to `theColor`.

window.setColor(Color.red);

method call with parameter

parameters

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



The diagram consists of four red arrows pointing upwards from the arguments of a method call to the parameters of a method signature. The first arrow points from '10' to 'int x', the second from '50' to 'int y', the third from '30' to 'int width', and the fourth from '70' to 'int height'. Each arrow starts with a red dot at the argument position.

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

method call with parameters

parameters

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

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

Four red arrows point from the arguments in the function call to the parameters in the function signature. The first arrow points from '10' to 'int x', the second from '50' to 'int y', the third from '30' to 'int width', and the fourth from '70' to 'int height'.

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.**

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

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.

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

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.

passbyvalueone.java

Passing by Value

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

OUTPUT

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

//test code

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

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.

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]
```


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.

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**

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();  
    }  
}
```



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.

passbyvaluethree.java

Start work on Lab 17