1. Log in
2. Unit 4
   - Writing Methods
   - Recursion
   - **Classes, Methods, and Objects**
   - **Scope and References**

Mar 9-11:28 AM

Scope of variables & methods ...

Variables are limited to the block { } they are declared in ...

```
int you=5, a=10;                                   //Global Variable

if(a==0)  {
     int me=8;                                     //Local Variable
     you = me+1;
}

System.out.println("you = "+you);                  //all good!
System.out.println("me =  "+me);                   //error!
```

Nov 9-7:48 PM

Scope of variables & methods ...

Variables are limited to the block { } they are declared in ...

```
public class tacos {

        String type="";                //instance variables - NOT STATIC
        int number=0;                  //they are used for specific instances

public tacos(String t, int num){
        type=t;
        number=num;
}
public static void main(String[] args) {
        tacos order1 = new tacos("hard",6);
        tacos order2 = new tacos("soft",12);
//      number=5;                      //illegal, not static, can't use in static main!
        System.out.println(order1.type);
        System.out.println(order2.number);
}
```

Nov 9-7:48 PM

You Can Use the 'this' keyword too ...

```
public int getNumber( )  {

        return number;                        //returns the instance's number
}

public int getNumber( )  {

        return this.number;                   //returns ... look at this instance's number
}

System.out.println(order1.getNumber( ));     // order1 = instance name - "passed"
```

Nov 9-7:48 PM

A better use of the 'this' keyword ...

```
public void printOrder( ) {

        System.out.println(this.number);       //print number for this instance

        System.out.println(this.type);         //print type for this instance

}

public static void main(String[ ] args) {

        tacos order2 = new tacos("soft",12);

        order2.printOrder();

}
```

Nov 9-7:48 PM

A common use for 'this'

```
public class tacos {
             String type="";
             int number=0;
        public tacos(String t, int num){
             type=t;
             number=num;
        }
```

Do you notice that we have typically used different names for the instance variables and the parameters?

Nov 9-7:48 PM

A common use for 'this'

```
public class tacos {
          String type;
          int number;
     public tacos(String t, int num){
          type=t;
          number=num;
     }
}
```

Do you notice that we have typically used different names for the instance variables and the parameters?

```
public class tacos {
          String type;
          int number;
     public tacos(String type, int number) {
          this.type=type;
          this.number=number;
     }
```

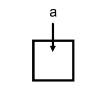
We are allowed to use the same instance and parameter names IF we specifically state to go to 'this' instance's state variable.

Nov 9-7:48 PM

References to variables and memory allocation ...

Primitive data types (int, double, String, boolean)
...has own memory!

int a;



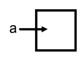
Memory is allocated for an integer. Variable name 'a' now points to an empty memory slot.

Nov 9-7:48 PM

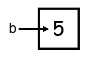
References to variables and memory allocation ...

Primitive data types (int, double, String, boolean)

int a;



int b=5;



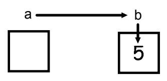
Memory is allocated for an integer. Variable name 'b' now points to the value 5 in memory.

Nov 9-7:48 PM

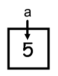
References to variables and memory allocation ...

Primitive data types (int, double, String, boolean)

a = b;



'a' will now equal whatever 'b' points to. 'b' points to 5, so a=5 (a is the value 5)



Fill a with whatever b is pointing to ...

Nov 9-7:48 PM

References to variables and memory allocation ...

Reference data types (int, double, String, boolean)
... refers to memory!

tacos order3 = new tacos("soft",50);;

*** Memory allocation is very different ***

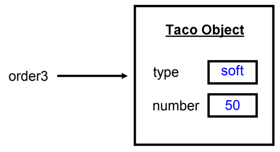
*** We are not creating a single memory slot ***

Nov 9-7:48 PM

References to variables and memory allocation ...

Reference data types (int, double, String, boolean)
... refers to memory!

tacos order3 = new tacos("soft",50);



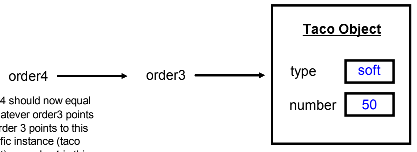
order3 is an instance of a taco object and therefore is a reference (or pointer) to a taco object.

Nov 9-7:48 PM

References to variables and memory allocation ...

Reference data types (int, double, String, boolean)
... refers to memory!

```
tacos order3 = new tacos("soft",50);
tacos order4 = order 3;
```

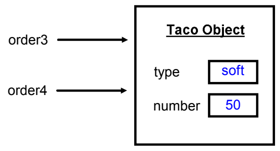


order4 should now equal to whatever order3 points to. order 3 points to this specific instance (taco object), so order 4 is this specific instance!

Nov 9-7:48 PM

IN FACT, HERE'S WHAT IS HAPPENING ...

```
tacos order3 = new tacos("soft",50);
tacos order4 = order 3;
```



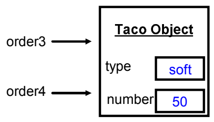
*** Can you see that this could be a major issue? ***

Nov 9-7:48 PM

This code should show the big issue here ...

```
public class tacos {
    String type;
    int number;
    public tacos(String type, int number){
        this.type=type;
        this.number=number;
    }
    public static void main(String[] args) {
        tacos order3 = new tacos("soft",50);
        System.out.println("Order 3 = "+order3.number+" "+order3.type+" tacos.");
        tacos order4 = order3;
        System.out.println("Order 4 = "+order4.number+" "+order4.type+" tacos.");
        order4.number=10;
        System.out.println("Order 3 = "+order3.number+" "+order3.type+" tacos.");
        System.out.println("Order 4 = "+order4.number+" "+order4.type+" tacos.");
    }

}
```



Nov 9-7:48 PM

Moral of the story ...

If you want a new object ... use 'new'!

~~tacos order3 = new tacos("soft",50);~~
~~tacos order4 = order 3;~~

```
tacos order3 = new tacos("soft",50);
tacos order4 = new tacos("soft",10);
```

Nov 10-9:26 AM

Now, let's talk about 'null' pointers ...

null - simply means "empty"

```
tacos order3 = new tacos("soft",50);
tacos order5;
```

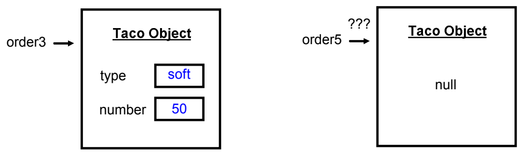
*** tacos order5 says we have a tacos object called order5 ***
*** It is a null pointer though since it never is told to point to something! ***

Nov 10-9:26 AM

Now, let's talk about 'null' pointers ...

null - simply means "empty"

```
tacos order3 = new tacos("soft",50);
tacos order5;
```



Nov 10-9:26 AM

Adding this code ...

```
order3=null;
System.out.println("Order 3 = "+order3.number+" "+order3.type+" tacos.");
```

Nov 10-9:37 AM

Adding this code ...

```
order3=null;
System.out.println("Order 3 = "+order3.number+" "+order3.type+" tacos.");
```

Will Create ...

the *NullPointerException* Error!

Nov 10-9:37 AM

Things to do ...

1. Be wrapping up all Unit 04 WS01-08 Worksheets
2. Exam coming up soon!

Nov 6-3:25 PM

Code used today

```
package unit4;
public class tacos {
   String type;
   int number;
   public tacos(String type, int number){
      this.type=type;
      this.number=number;
   }
   public tacos(){
      type="";
      number=0;
   }
   public static void main(String[] args) {
      tacos order3 = new tacos("soft",50);
      System.out.println("Order 3 = "+order3.number+" "+order3.type+" tacos.");
      tacos order4 = order3;
      System.out.println("Order 4 = "+order4.number+" "+order4.type+" tacos.");
      order4.number=10;
      System.out.println("Order 3 = "+order3.number+" "+order3.type+" tacos.");
      System.out.println("Order 4 = "+order4.number+" "+order4.type+" tacos.");
      tacos order5=new tacos();
      System.out.println("Order 5 = "+order5.number+" "+order5.type+" tacos.");
      order3=null;
      System.out.println("Order 3 = "+order3.number+" "+order3.type+" tacos.");
   }
}
```

Nov 10-9:42 AM