The contents of this document represent and go over the topics covered in lecture on 2/28 and 3/09 Use this to review the content as you please, also be sure not to use these in place of lecture as these documents will only cover lecture highlights and important details and do not represent the full scope of what you may be tested on. As always if you have any questions ask on ED!

Lecture 13 - 2/28/2023

All you learned about today was method overloading and you revisited the this keyword. Here is a quick review of that.

To overload a method (or a constructor for that matter) you are simply going to create a method with the same name as the method you wish to overload and you will change the rest of the method signature that is you can do one of two things:

1. Change the number of arguments
2. Change the data types of the arguments

Take the following method definition for a method that returns the sum of its arguments:

```
public abstract int addArguments(int a, int b);
```

Note don't worry about the keyword abstract, I just wrote it like that so my method definition would be syntactically correct ending with a semicolon.

Anyway, the two of the following are acceptable overloads for the addArguments method and one of them isn't try to guess which it is before I give away the answer:

```
public abstract int addArguments(double a, double b);
public abstract void addArguments(int a, int b);
public abstract int addArguments(int a, int b, int c);
```

The first and third are valid overloads, you cannot overload a method by just changing the return type of the method.

Finally we revisited the this keyword, which we used to create a method to transfer money from one BankAccount object to another, a reminder of the method:

```
public void transfer(BankAccount other, double amt){
     this.withdraw(amt);
     other.deposit(amt);
}
```

Here the this keyword refers to the current or executing object, while other is trivially the reference to the BankAccount object passed into the method. You also saw how you could use the this keyword in constructors to avoid ambiguity when it comes to initializing your instance variables.

Lecture 16 - 3/09/2023

Today was all about Arrays! Well Arrays and Command Line arguments.

Arrays are collections of items of a single type. You can have int arrays, String arrays and BankAccount object arrays and an array of pretty much any type, including an array of arrays!

To declare an array - the example I talk about will revolve around ints - you do it exactly how you would for anything else except we will use square brackets to denote the array. So for an int array I would declare it as:

```
int[] myFirstArray;
```

To initialize the array you have one of two options:

1. Initialize the Array with default values
2. Initialize the Array with just a length

This is what each of those options would look like:

```
myFirstArray = {1,0,0,4};
myFirstArray = new int[4];
```

The first way initialized my array with 4 values, the second way initialized my array with just a length - in this case 4 - in Java int arrays receive the default value of 0. So the second option is equivalent to:

```
myFirstArray = {0,0,0,0};
```

Now that we know about arrays we can learn how to iterate through them. When using a for-loop we can loop through an array as follows:

```
for(int i = 0; i < myFirstArray.length; i++){
                //code to run
 }
```

An important trait to notice is that we did not write myFirstArray.length() but rather the former without the parentheses. This indicates that length is not a method for an array but rather a variable, but in this case a constant, that is to say arrays are immutable in Java, once they have a certain size initialized with them they cannot change size.

Now we know that whenever we wrote a main method we wrote it with a single argument, String[] args that is a String array named args, using the looping technique above we can print out the contents:

```
for(int i = 0; i < args.length; i++){
     System.out.println(args[i]);
}
```

We access elements of an array using the bracket notation with an integer from 0 to length-1 within them. Finally to wrap up to provide arguments to args we simply write them when we execute our Java file:

```
java SomeJavaClass these are command line arguments
```

The above code snippet would print:

```
these
are
command
line
arguments
```

You also talked about static variables. Static variables are variables that keep a copy per class NOT per object like instance variables. This means if we have 3 instances of a class with a static variable a change to one of them affects the others. That is the gist of it at least.

Have a wonderful Spring Break! If you are not doing anything and want to improve your Java skills, I will be hosting Java Review Sessions focused mainly on the Java aspect of it all!