Lecture 15 - 3/5/2024

Today is the day we learn about arrays, and by the end of lecture you will be fully equipped to understand the following line:

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

We begin with a discussion on what types you can put in an array, which is any type that you have in java. Arrays are also homogenous, meaning all the types contained within have to be the same. Arrays are also immutable like Strings, you can change the contents of an array however you want.

Think of it like this, treat an array like a bookshelf, you can change the things on the bookshelf without changing the bookshelf itself.

Consider the following ways to instantiate an array:

1. `int[] numbers = {1,2,3,4};`
2. `int[] numbers = new int[]{1,2,3,4};`
3. `int[] numbers = new int[4];`

The first 2 ways provide default values while the last way does not have any predefined values from the user. Arrays do have default values associated with their positions based on the type, for integral values it’s 0. For objects it’s null, for floating points it's 0.0, for chars it's the null character, and for booleans it's false.

Now we can go back to args, which is just an array of Strings. We enter the values into args at the command line. Consider the following terminal lines (from codio lecture 15):

```
javac ArgsStrings.java
java ArgsStrings Happy Days are Here!
```

Clicking enter would run the ArgsStrings class’s main method which prints out all of the values of args.

Lecture 16 - 3/7/2024

We began exploring a new archetype in lecture, now we have AudioBook along with BankAccount. We use AudioBook and AudioBookCollection to demonstrate different functionalities in Java primarily with arrays. We discussed further about what it means for an object reference to be pointing at something. Consider the following example:

```
private void increaseSize() {
  AudioBook[] temp = new AudioBook[collection.length * 2];

  for(int i = 0; i < collection.length; i++)
    temp[i] = collection[i];

  collection = temp;
}
```

What this method does is copy over the contents of collection to a new array called temp. We then change collection reference to now point to where temp points.

We start with a single array object - collection - and within it are 100 object references to null. Whenever we add an AudioBook we replace one of those references to null with a reference to an AudioBook.

In this method, we make a new variable called temp of type AudioBook[] which will have a length twice the size of the current collection array. Then we loop through our collection and assign each of the first collection.length slots of temp to the preexisting AudioBook references in collection. The rest will remain null. Finally we make collection point to this new expanded array temp.

You were also taught about the Comparable Interface, which allows us to compare AudioBooks to one another. An interface acts as a contract in Java stating that you will provide an implementation of any method that is in the interface. For Comparable this is the compareTo method.

When given a.compareTo(b):

- We return -1 if a < b (a comes before b)
- We return 1 if a > b (a comes after b)

   - We return 0 if a == b (a and b are the same)

This allows us to write sorting algorithms/sort custom objects that we write.