Introduction
Last time we discussed all things related to Computer Hardware and Binary, if you need a refresher on any of those topics please go view “Notes Session 3” Today we will talk about Class Design, Arrays, and ArrayLists so let’s jump straight into it.

Class Design Overview
Class design is a very important aspect of programming as it allows us to organize and properly think out how we want to write our code before we even touch the keyboard. When I was in high school taking my first Computer Science course, my instructor told me to think twice and code once, and that is the mindset I want you to have going forward.

There are two key design principles that you should be aware of: Top-Down Design and Bottom-Up Design I will now give a brief description and advantages of each principle

Top-Down Design
In the top-down model, an overview of the system is formulated without going into detail for any part of it. Each part is then refined into more details, defining it in yet more details until the entire specification is detailed enough to validate the model. In top down design we perform the following actions:

- We break the problem into parts,
- Then break the parts into parts soon and now each of the parts will be easy to do.

Some of the key advantages of Top-Down Design are:

- Breaking problems into parts help us to identify what needs to be done.
- At each step of refinement, new parts will become less complex and therefore easier to solve.
- Parts of the solution may turn out to be reusable.
- Breaking problems into parts allows more than one person to solve the problem.

Bottom-Up Design

In this design, individual parts of the system are specified in detail. The parts are linked to form larger components, which are in turn linked until a complete system is formed.

Some of the key advantages of Bottom-Up Design are:

- You know each individual part works
- Redundancy is minimized by using data encapsulation and data hiding.

Java Specific Design

Regardless of which design methodology you decide to go with for your project there are some java specific design principles that you should follow, most of these are already laid out in the style guide on courseworks but this will be a brief overview

Be sure that all .java files contain a header at the top of the file detailing the author's name, the date, the filename, and a brief description of what the file's functionality is. For university students it is important to put your unique identification tag

When you open code blocks you use the '{' to denote the beginning of a block and the '}' however there are two key styles that you should follow when writing your code, I will detail each below but please choose one style and stick with it through the entire file to keep the file readable and to maintain consistency.

K&R Style (my personal preference) braces go attached to the preceding line:

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

Horstmann Style braces are detached from the preceding line:

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

Indentation and whitespace are important components of a program, if you are writing in a language like Python then you are required to have proper indentation and whitespace in order for the program to run in the first place. In java it is purely style, but style is important so it is relevant, use whitespace to separate ‘thoughts’ in your code so if you have 5 lines devoted to setting up a single variable put a white space afterwards to improve readability. For every codeblock you are deep into a program, there is one additional indentation that is needed.

Class names are written in UpperCamelCase; Variable and Method names are written in lowerCamelCase, please provide useful and informative names.

Constants in Java should be written in CONSTANT_CASE

All instance variables should have the private access modifier on them and have corresponding mutator and accessor methods as you deem necessary for the proper encapsulation of data and program functionality.

All methods that you want the client to be able to call should have the public access modifier, if you do not want the client to have permission to call the method directly (for example helper methods) they should have the private access modifier.

Introduction to Arrays

An array in Java is a group of like-typed variables referred to by a common name. Arrays in Java work differently than they do in C/C++. Following are some important points about Java arrays.

- In Java, all arrays are dynamically allocated. (discussed below)
- Since arrays are objects in Java, we can find their length using the object property length. This is different from C/C++, where we find length using sizeof.
- A Java array variable can also be declared like other variables with [] after the data type.
- The variables in the array are ordered, and each has an index beginning from 0.
- Java arrays can also be used as a static field, a local variable, or a method parameter.

- The size of an array must be specified by int or short value and not long.
- The direct superclass of an array type is Object.

An array can contain primitives (int, char, etc.) and object (or non-primitive) references of a class depending on the definition of the array. In the case of primitive data types, the actual values are stored in contiguous memory locations. In the case of class objects, the actual objects are stored in a heap segment (heaps are not relevant information to you at this time so that is all that will be said).
To declare and instantiate an array in java it can be done in one of two ways

```
int[] arr = new int[5];
int arr[] = {0,1,2,3,4};
```

It is important to note that the [] can be placed after the type or after the name. It makes no difference, the first method to instantiate declares an int array that can hold 5 elements, the second method does the exact same thing except it gives initial values to each slot in the array. An important thing to be able to do with arrays is to iterate through them we can do this one of two ways:
The first is with a traditional for-loop:

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

Using a traditional for-loop allows us to modify the data structure without running into errors. You can access elements at index i with arrayName[i] syntax.
The second way is with an enhanced for-loop otherwise known as a for-each loop

```
for(int i : arr){
  System.out.println(i);
}
```

When using the enhanced for-loop be careful not to modify the structure of the data structure you should only be accessing the elements using a for-each loop

To add items to the array you just specify the index in assignment:

```
arr[2] = 5;
```

This line means that we are assigning the value 5 to index 2 of the array

Introduction to ArrayLists

Now that we have covered arrays, which are static, we need to address the possibility that we may have an unknown amount of data, so an array which cannot change length past the initial allocation, an ArrayList can solve this issue because it is dynamic and can expand when needed.

The main caveat with ArrayLists is that primitive data types cannot be stored within the structure and the autoboxed object types must be used (we will cover this next time)

To declare and instantiate an ArrayList, you must have imported the java.util.ArrayList package and then you write the following style of code:

```
ArrayList<Integer> myArrayList = new ArrayList<Integer>();
```

The usage of 'Integer' on the right hand side is redundant and you actually do not need it meaning the following line of code is just as valid:

```
ArrayList<Integer> myArrayList = new ArrayList<>();
```

ArrayLists have dedicated methods for adding, removing, getting and setting elements within the Array, please go see the Important Classes and Methods document in the files section on Courseworks.

You can iterate through ArrayLists the same way as with arrays except you'd use the proper methods of the ArrayList class rather than the square bracket syntax style like you would for Arrays.

While I will not detail how the methods work here, the methods that are most important from the ArrayList class are: add(), get(), set(), and remove()

Array vs ArrayList

Here are some key things to keep in mind when you are deciding whether to use an Array or an ArrayList for your program:

1. Flexibility
    a. Arrays are static data structures
    b. ArrayLists are dynamic data structures

2. Primitive Data Type
    a. Arrays can store primitive data types and object types
    b. ArrayLists can only store object types

3. Adding Elements
    a. Arrays can insert elements via assignment
    b. ArrayLists can add elements via methods such as add()

4. Dimensionality
    a. Arrays can be multidimensional if you need them to be
    b. ArrayLists are only in one dimension

Conclusion

This notes session provided discussion about Class Design Patterns, and introductions to Arrays and ArrayLists. Next time I will talk about higher dimensional arrays, Autoboxing types, Inheritance and Introduce Exceptions