Representing Information

Computers at the lowest level store information in a single bit 8 of these bits makes a byte. We can represent these strings of bits as binary numbers. We refer to binary numbers as “Base 2” numbers since the individual digits can only have one of two values 0 or 1, just like the counting system we use traditionally is “Base 10” since our digits can be one of 10 values 0-9.

We can convert between these values with relative ease by exploiting the fact that we know that starting from the right hand side of the number has a placeholder value of 10^0 or 2^0 if we are in binary. With the power increasing by 1 with every shift to the left.

We also know that there is another counting system known as Hexadecimal also known as “Base 16”. There are a couple problems on the practice exam that deal with converting between these bases.

More Object Oriented Programming

We didn’t learn much new coding techniques between the first and second midterms but we became better at writing the code we already knew. We know more about parameter passing and how to count objects and object references and what it means for an object to be eligible for garbage collection. We will go over in more detail what all of this means.

In Java we pass parameters by value, NOT by reference, this means for primitive types we get a local copy of the value passed into the method, any change to this local value does not affect the original value passed in. For Objects however, the “value” being passed in is the memory location of the Object. What this means is any mutation we perform to the object in the method is reflected in the object that was passed into the method.

When counting objects and object references it is important to know that there can be multiple object references for a single object. A new object is created anytime we use the new keyword. Object references pointing to the same object in memory are done through simple assignment. An object is eligible for garbage collection as soon as there are no more references pointing to that object.

Arrays and ArrayLists

Recall that we can declare and initialize an array in Java in the following way:

```
 DataType[] variableName = new DataType[anIntRepresentingLength];
```

So applying this to ints we get the following piece of Java code:

```
                  int[] intArray = new int[5];
```

This creates a new int array that has 5 slots to hold elements, unless you provide initial values then the elements inside the array will receive their default values which for ints would be 0 and for any ObjectType it would be null.

For ArrayLists we first need to make sure we import the necessary package prior to attempting to use the Data Structure. We can declare and instantiate an ArrayList as follows:

```
     ArrayList<ObjectType> variableName = new ArrayList<>();
```

We can access the values of an array by using the bracket notation along with the specific index we wish to access so you'd write things like intArray[0] and intArray[intArray.length-1] for an ArrayList we can access elements using the get() method with the sole argument being the index we wish to access so the equivalent would be variableName.get(0) and variableName.get(variableName.size()-1). Important time to note that for ArrayLists we use the size() method to return the size of the ArrayList since it is able to dynamically change its size unlike an array which has a fixed length.

Please see the Important Classes and Methods pdf for more information on these methods and other methods you should know. Here is just a brief overview of the methods that are commonly used for an ArrayList:

1. add()
2. set()
3. get()
4. size()
5. remove()