Lab 4 – Jonathan Luu

**Part 1 – Collections of People**

I created a Person class with three instance variables: name, email, and birthday.

In my ListOfPeople class, I played around with ArrayList and Linked Lists.

ArrayList

* I created 6 people and added them to the array list
* I iterated through this list and printed their information by using the printList() method and overriding the toString() method in the Person class
* The printList() method takes in two parameters – the list of people and a String describing what the list is. This method is used to print out either all the people in the list or a sub-list of people. This method also takes advantage of polymorphism as its parameter is a List, allowing it to print both LinkedList and ArrayList containers
* The method printSubArrayList() utilizes the ArrayList subList method, which takes in a range to create a subset of that array list. It then passes this new subset into printList()
* The method arrayListB finds all the people who’s name starts with the letter ‘B’ and adds them to a new ArrayList. It then returns this list.

LinkedList

* I created 6 people and added them to the linked list
* I iterated through this list and printed their information by using the printList() method
* I then used the peek() function to get the first and last people from the list and printed them out
* I popped some members from the end of the list and then used the peek() function again to verify it had changed

Comparison

* Both lists have similar operations – add, remove, size – but LinkedList has remnants of a traditional link data structure. LinkedList has some repetitive methods, such as add and push, remove and pop, and it also has getFirst, getLast, and peek methods which emphasize its benefit for adding and removing elements, especially at the front or end of the list.
* ArrayList does not have all those extra methods and instead has only the typical methods of add, remove, size, and set.
* LinkedLists are typically worse at finding a position in the list as you have to iterate through it, while arrays allow for fast random access when you have to find a position. However, arraylists have to resize themselves if we do not call the ensureCapacity function to accommodate when necessary, while linkedlists just add to the tail of their list by updating pointers.
* Although there are some runtime differences between the two, when experimenting with the two lists in this lab without using timers, they felt very similar due to the pre-implemented nature of the collections.

In my MapOfPeople class, I played around with HashMaps and TreeMaps

* I performed similar tasks done in the List part – adding objects, iterating through the map, finding objects/keys – but also did new tasks such as getting all the keys from the map and getting all the objects from a map returned as a set. I also learned about the existence of TreeMap which I had never used before and its various functions. One particular function that I used was tailMap(), which takes in a key value and returns a SortedMap of <Key,Value> pairs that have a greater than or equal to key-value of the input. Additionally, most of the TreeMap functions were new to me as well as I had only used HashMap in the past.
* I chose to make the Key value the person’s name, and the Value of the map was the person object. By doing so, I do not believe I have to change the Person class as it is the second parameter. However, if I made Person the key value, I would have to implement a compare function so that the Map can sort the values accordingly and utilize its functions, a majority of which rely on sorting. As I used a String value as my key, its compare function was already implemented so nothing new had to be added.