1. Register your attendance.

2. Go to Groups on Wide and copy the folder CSCU9Y4/Practicals/Practical3 into your Y4 Practicals folder.

   The aim of this practical is to give you practice in manipulating stacks and queues, and also to make you think about the difference between ArrayList and LinkedList packages.

3. In the CoffeeShop example from practical 1 you used a queue of customers. Here, instead of simply using the library Queue operations directly, implement the Queue operations using the underlying List methods. (Recall, a Queue is FIFO: elements are added at one end and removed from the other end.) The outline code is given to you in QueueSkeleton in Practical3. Implement with the type parameter K.

   ```java
   class Queue<K> {
       private List<K> elements;

       public Queue() { ... }
       public int howMany() { ... }
       public void push(K k) { ... }
       public K pop() { ... }
   }
   ```

4. Test your implementation to be sure it works correctly. This is most convenient using an Integer to instantiate K (it makes it easy to generate elements of the queue in a loop). You will need to demonstrate these tests for the checkpoint. Be sure to cover edge cases (for example, what happens when the queue is empty?).

5. Did you implement List as an ArrayList or a LinkedList? Why? (write the answers here)

6. What if we wanted a stack? Repeat the exercise in parts 3 and 4, this time implementing a Stack<K>. Define a new class - don’t overwrite your Queue - you need it for the checkpoint.
7. Did you stick with your choice of underlying implementation (ArrayList or LinkedList)? If not, why not? And if you did, are there any efficiency considerations? Test your answer by implementing the alternative and adding a large number of items to the queue/stack.

(write your answer here)

Checkpoint

Now demonstrate to a tutor that you have completed all this week’s tasks. You should show us StackQueue with the methods for each data structure, and your answers to the questions about ArrayList vs LinkedList.

If you didn’t get to this point, finish the work off during the coming week and get the checkpoint marked next time.

Need something more to do? Here are a couple of optional extras to give practice with iterators:

Assume a list vals1 containing a list of Integer objects. Using an iterator, write a method:

```java
public void noNegs(List<Integer> vals) { ... }
```
in which all negative values in the list are replaced by zero.

Assume an array vals2 of objects, some of which may be duplicates. We want to replace this with an array in which each object is not equal to any other object in the array. Will the following code do the job? Why?

```java
Object[] vals2 = new Object[50];
// code to put values in the array
List aList = new ArrayList(Arrays.asList(vals2));
Set aSet = new HashSet(aList);
vals2 = aSet.toArray();
```