Solution --- Lecture 8.3 Programming Exercise

This is a possible solution to the programming exercise. Your solution does not have to be exactly the same, but it should produce similar results.

```
import java.util.*;

public class QueueExercise
{
      public static void main( String [] args )
      {
            LinkedList< Customer > queue = new LinkedList< Customer >();
            char response = 'a';
            String nm = " ";
            int time = 0;


            Random rn = new Random( 20101119 );       // create a rn generator

            while ( response != 'q' )
            {
                  nm = Input.getString( "Enter a customer name" );
                  time = rn.nextInt( 20 ) + 1;

                  queue.addLast( new Customer( nm, time ) );
                  response = Input.getChar( "Enter 'a' to add a customer,
                                            'q' to quit" );
            }

            System.out.println();
            for ( Customer c : queue )
                  System.out.println( "Name is " + c.getName() +
                                      "  Service time is " + c.getTime() );
      }
}
```

The Customer class is defined as follows:

```
public class Customer
{
	private String name;
	private int serviceTime;

	public Customer( String n, int t )
	{
		name = n;
		serviceTime = t;
	}

	public String getName()
	{
		return name;
	}

	public int getTime()
	{
		return serviceTime;
	}
}
```

A second solution is also provided. In this solution we have defined a new generic class called FIFOQueue that is based upon the LinkedList class. The FIFOQueue class is used to model the basic behaviors of a FIFO queue…one can add an object to the end of the queue, get the object at the head of the queue, and get the size of the queue.

```
import java.util.*;

public class FIFOQueue< T >
{
	private LinkedList< T > list = new LinkedList< T >();

	public void add( T item )
	{
		list.addLast( item );
	}

	public T get()
	{
		return list.removeFirst();
	}

	public int size()
	{
		return list.size();
	}
}
```

Now, the solution is almost identical to the first sample solution except that a FIFOQueue class is used and a while() loop is used in place of the collection-based for loop because the collection-based for is not defined for the FIFOQueue class.

```
import java.util.*;

public class QueueExercise2
{
	public static void main( String [] args )
	{
		FIFOQueue< Customer > queue = new FIFOQueue< Customer >();
		char response = 'a';
		String nm = " ";
		int time = 0;


		Random rn = new Random( 20101119 );
		while ( response != 'q' )
		{
			nm = Input.getString( "Enter a customer name" );
			time = rn.nextInt( 20 ) + 1;

			queue.add( new Customer( nm, time ) );
			response = Input.getChar( "Enter 'a' to add a customer,
                                    'q' to quit" );
		}

		System.out.println();
		while ( queue.size() > 0 )
		{
			Customer c = queue.get();
			System.out.println( "Name is " + c.getName() +
                            "  Service time is " + c.getTime() );
		}
	}
}
```