CMPS 303 Data structures
Case study: The marines and cannibals problem

Nine shipwrecked mariners found themselves on an island populated by cannibals who can count only five by five. To make their selection of the victims easier the cannibals place the mariners on a circle, starting from one position chosen randomly, they count 5 and eat the mariner located at new position, then the process is repeated. According to the tradition, the last shipwrecked mariner should be released (will not be eaten). e.g. [1 2 3 4 5 6 7 8 9] if they start from 2, then the mariner at position 6 will be eaten, the circle becomes [1 2 3 4 5 **7** 8 9], the second round will start from position number 6, hence mariner 2 will be eaten,...etc. If they start from 8, then the mariners 3, 8, 5, 2, 1, 4, 7, and then 9 will be eaten consecutively. Mariner 6 is the lucky person and will not be eaten.
Use a circular list to simulate this process.

SOLUTION

```
// node class
class node {
        public int val ;
        public node next ;

        node(int v) {
                val= v ;
        }
}

// list class
class marinersList {
        public node head;

        public marinersList() {
                head = null;
        }

        public node getHead() {
                return head ;
        }

        public String toString() {
                String s="" ;
                node cur = head ;
                while(cur.next !=head) {
                        s+="["+cur.val+"]"+"->" ;
                        cur = cur.next ;
                }
                s+="["+cur.val+"]" ;
                return s ;
        }

        public boolean isEmpty() {
                return head == null ;
        }
```

```
        // on the top of the list
        public void addMariner(int v) {
                node myNew = new node(v) ;

                // if list empty
                if(isEmpty()) {
                        myNew.next = myNew ;
                }
                else {//find the last node
                        node n = head ;
                        while(n.next!=head) n = n.next ;
                        n.next = myNew ;
                        myNew.next = head ;
                }
                head = myNew ;
        }

        public void eatMariner(int v) {
                node cur= head ;
                node pred=  head ;
                //find v
                while(cur.val!=v) {
                        pred= cur ;
                        cur= cur.next ;
                }

                // if we remove the first node
                if(cur==head) {
                        //find the last node
                        node n = head ;
                        while(n.next!=head) n = n.next ;
                        n.next = head.next ;
                        head= head.next ;
                }
                else {
                        pred.next = cur.next ;
                }
        }
}

class marinersListApp {
        public static void main(String[] args) {
                marinersList ml = new marinersList() ;
                ml.addMariner(9);
                ml.addMariner(8);
                ml.addMariner(7);
                ml.addMariner(6);
                ml.addMariner(5);
                ml.addMariner(4);
                ml.addMariner(3);
                ml.addMariner(2);
                ml.addMariner(1);

                Random rn = new Random();
                int firstPos = rn.nextInt(9) + 1;
                System.out.println("Count 5 by 5 starting from mariner "+firstPos);

                // Find eatFirst
```

```
                node n= ml.getHead() ;
                for(int i=0; i<firstPos-1; i++) n= n.next;

                // Find those to be eaten
                node pred = null ;
                for (int mar=0 ; mar<(9-1) ; mar++){
                // print alive mariners
                System.out.println(ml);
                for (int lg=0; lg<5; lg++){ pred = n ; n = n.next ;}
                System.out.println("Mariner "+pred.val+" has been eaten");
                ml.eatMariner(pred.val);
        }
        // print the last living mariner
        System.out.println(ml+" -> I am the only mariner who remains alive !");
}
```