OODP 2013 — Assignment One

What to Write

1. First, complete the `List` class partially defined as:

```
package assignments.ass1;

/*
 *  List class
 *
 *  A List is an ordered collection of any kind of object specified in the parameter.
 *
 *  Operations:
 *     addToEnd    Add a given object to the end of the list.
 *     toString    A String form of the objects in the list in order,
 *                 enclosed in square brackets, separated by spaces.
 */
public class List<T> {
    private static final int INIT_LEN = 10;
    private T[] items; // the actual items
    private int numItems; // the number of items currently in the list

    /*
     * constructor: initialize the list to be empty
     */
    @SuppressWarnings("unchecked")
    public List() {
        items = (T[]) new Object[INIT_LEN];
        numItems = 0;
    }

    /*
     * AddToEnd
     *
     * Given: Object obj Do: Add obj to the end of the list.
     */
    public void addToEnd(T obj) {}

    /*
     * toString
     *
     * A String form of the objects in the list in order, enclosed in
     * square brackets, separated by spaces.
     */
    @Override
    public String toString() {
        return null; // REPLACE WITH YOUR CODE
    }
}
```

   (a) Write the body of the `addToEnd` method, which should add the given object to the

end of the list.
Note that the `items` array might be full. In that case, you should allocate a new array that is twice the size of the current one and copy the old objects into the new array before adding the new object.

(b) Write the body of the `toString` method, which should return a `String` consisting of a left square bracket followed by each object in the list, separated by spaces, followed by a right square bracket. (If the list is empty, just return `[]`.)

(c) Write a `main` method to test your `List` class (do not make the main method part of the `List` class).
Be sure to test *boundary* cases (e.g., calling the `List` methods when the list is empty) as well as other cases.

2. Now add additional fields and methods to allow a client of the `List` class to iterate through the list, accessing each item in the list, as follows:

(a) Add a `currentObject` field to the `List` class that is (conceptually) a pointer to the current object. (In fact, you should implement this field using an integer whose value is the array index of the current object.)

(b) Add a method `firstElement`, which makes the first object on the list be the current object (i.e., set the `currentObject` field to zero).

(c) Add a method `nextElement`, which returns the current object, and also updates the `currentObject` field to *point to* the next object in the list.
Note that it doesn't make sense to call this method when the current-object *pointer* has *fallen off* the end of the list or when there are no elements in the list.
For now, you can just assume that the method will not be called in those cases; you will learn how to handle those cases using an exception later.

(d) Add a method `hasMoreElements`, which returns `true` if the list is not empty and the current-object *pointer* hasn't *fallen off* the end of the list (i.e., if there are still more items in the list that haven't been accessed yet).
Note that this method should return `true` when the current-object *pointer* is pointing to the last object in the list — it should only return `false` when the list is empty or the current-object pointer has *fallen off* the end of the list.

(e) Update your `main` method to test the three new methods.

To help you understand these methods better, look at the following `main` method, which first creates a list of strings, then prints each string:

```
public static void main(String[] args){
  // create a list of Strings
    List<String> l = new List<>();
    l.addToEnd("Fred");
    l.addToEnd("Betty");
    l.addToEnd("Judith");
    System.out.println(l);
}
```

What to submit

Put all of your code in a single file called `List.java`, and submit your code in your portfolio and do not submit any files other than `List.java` (for example, do not submit `List.class`).