CSC 151 Assignment 8

Objectives:

- Implementing solutions that uses java.util
- Applying recursion for packing and unpacking dynamic lists

Definition

In a list, there can be repeated items and they can be represented in a more efficient way if they are consecutive. Run-length encoding/decoding method can help us to compress/uncompress these lists. Your task will be implementing these methods. The packing method gets a list that has integer values and then finds the frequencies of each sequence. The method generates a new list where you have the pairs as the integers in the list and their frequencies instead of their repeated sequence. The unpacking method will reverse this operation, i.e., you will be given a packed list where you have pairs of integers and their frequencies and you will obtain the original list with the integers and their repeated sequences. There will be a helper method for unpacking an item. All methods will be **recursive.**

For example:

If you have a list

```
[1, 1, 1, 2, 3, 3, 3, 3, 3, 4, 4]
```

And you pack it, the result will be

```
[[1, 3], [2, 1], [3, 5], [4, 2]]
```

Since we have 1 that occurs 3 times first item will be [1, 3] and 2 that occurs once second item will be [2, 1] and 3 that occurs 5 times third item will be [3, 5], etc.

If you have a packed list

```
[[1, 3], [2, 1], [3, 5], [4, 2]]
```

And you unpack it, the result will be

```
[1, 1, 1, 2, 3, 3, 3, 3, 3, 4, 4]
```

Since we have 1 that occurs 3 times we add three 1's and 2 that occurs once we add a single 2 and and 3 that occurs 5 times we add five 3's, etc.

Class ListItem

- Use the partial code provided at the end of the document.
- It will be in **package assignment** (it is required otherwise **Gradescope** cannot run the tests)
- See the details below written as comments and complete the partial code as instructed.
- Test your class in the main method.
- Test your class by Junit

__Class ListEncodeDecode__

- Use the partial code provided at the end of the document.
- It will be in **package assignment** (it is required otherwise **Gradescope** cannot run the tests)
- See the details below written as comments and complete the partial code as instructed.
- **All methods will be recursive. Do not use iteration!!!**
- Test your class in the main method.
- Test your class by JUnit

__General Submission Instructions__

- You may have groups for the assignments **if it is allowed.** If you have a group state it **explicitly in the honor code** by writing all group members' names. Please name the pdf files accordingly. **However, all group members should submit individually.**
- All submissions require a package called **assignment** and your source codes (.java files) will be under it.
- You should submit
    - Your java files to **Gradescope** as required and
    - Your java files and sample outputs for various scenarios as a **single pdf** file named as **YourFullName(s)_AssignmentN.pdf** with the **honor code** at the top as a comment to Nexus under the appropriate link. Replace N in AssignmentN with the appropriate value. (e.g., if I submit Assignment1 I will submit a pdf file whose name is **ZeynepOrhan_Assignment1.pdf.** If I have a group whose members are me and Jane Doe, then the file name will be **JaneDoeZeynepOrhanAssignment1.pdf** and the honor code will be modified). Use the template .docx file attached and modify it accordingly.
- **If you use any resource to get help, please state it in the honor code. Otherwise, it will be an honor code violation!!!**
- **Gradescope provides a feature to check code similarity by comparing all submissions. If the percentage of your code's similarity is above a threshold, then this will be an honor code violation!!!**
- Make sure that you have no compiler errors.
- When you have compiler error free codes and submit appropriately to Gradescope, your code will be tested automatically and graded. You do not need to know the test cases, but you will see the test results. Informative messages will be provided for success and failure cases.
- You can resubmit your files any number of times if you have failing tests until the due date.
- Your final submission or the one you selected will be graded after the due date of the assignment.

- Please start as early as possible so that you may have time to come and ask if you have any problems.
- Read the late submission policy in the syllabus.
- Please be aware that the correctness, obeying the OOP design and submission rules are equally important
- Gradescope testing will generally be 100% and sometimes manual grading will be done. Please be aware that **your grade can be reduced if you do not follow the instructions and apply good programming practices**.
- Use the starter code if provided.
- Do not use any predefined Collection classes of Java unless stated otherwise.

Assignment specific instructions

- General submission instructions apply.
- **Group work: NOT ALLOWED!!!**
- **Due date:** February 27, 2021, until 11:50 PM
- **Nexus:** Submit the code file (**ListItem.java, ListEncodeDecode.java**) as a single pdf file named as **YourFullName_Assignment6.pdf** with the honor code at the top as a comment under Assignment 8 link.
- **Gradescope:** Submit the java files (**ListItem.java, ListEncodeDecode.java**) under the Assignment 8
- **Grading:**
    - Gradescope: Yes
    - Manual: No
- **Starter code:** Use the code given below

```
/*
 * Honor code
 */
package assignment;

/**
 * The class for an integer and frequency pair
 * @author Zeynep Orhan
 *
 */
public class ListItem {
	//Instance variables int item and int freq


	//Constructors: One with no parameters and one with 2 integer parameters


	//setters and getters


	//toString


	//hashCode


	//equals

	public static void main(String[] args) {
		ListItem lI1 = new ListItem(1,1);
		ListItem lI2 = new ListItem(3,5);
		ListItem lI3 = new ListItem(8,4);
		System.out.println(lI1);
		System.out.println(lI2);
		System.out.println(lI3);
	}

}
```

```
package assignment;

import java.util.*;

/**
 * This class packs/unpacks the lists of integers
 * The repeated items are represented
 * by the item and its frequency in the packed version
 *
 * The items and the frequencies can be unpacked
 * as the full representation
 *
 * @author Zeynep Orhan
 *
 */
public class ListEncodeDecode {

	/**
	 * Recursive static method packed: Pack the list of repeated sequences into item and frequency pairs
	 *
	 * @param repeatedList an ArrayList of Integers of repeated sequences
	 * @param packList an ArrayList of ListItem where we have item and frequencies
	 * @param start index of the starting position in the repeatedList
	 */
```

```
        /**
         * Recursive static method unpacked: Unpack the list of item and frequency pairs into repeated sequences
         *
         * @param packList an ArrayList of ListItem where we have item and frequencies
         * @param repeatedList an ArrayList of Integers of repeated sequences
         * @param start index of the starting position in the repeatedList
         */
```

```
        /**
         * Recursive static method unpackItem: Unpack one item, add item to repeatedList howMany times
         * @param item int item to be added
         * @param howMany int frequency of the item
         * @param repeatedList ArrayList<Integer> the result of adding item howMany times to this list
         */


        public static void main(String[] args) {
                 Integer items[] = {2};
                 //Integer items[] = {1,1,1,2,3,3,3,3,3,3,4,4};
                //Integer items[] = { 1, 1, 1 };

                ArrayList<Integer> uncompressed = new ArrayList<>();
                ArrayList<ListItem> compressed = new ArrayList<>();
                ArrayList<Integer> newUncompressed = new ArrayList<>();
                Collections.addAll(uncompressed, items);
                packed(uncompressed, compressed, 0);
                unpacked(compressed, newUncompressed, 0);
                System.out.println(uncompressed);
                System.out.println(compressed);
                System.out.print(newUncompressed);

        }

}
```

Output

```
[2]
[[2, 1]]
[2]
```