EECS1022 – Lab 7: ArrayList and HashMap

Fall 2025 **Due Date: Tuesday, December 2(11:59pm)**

Overview

Welcome to Lab 7!

This graded lab assignment focuses on Java Collection Framework, specifically ArrayList and HashMap data structures. Students will implement six utility methods that demonstrate proficiency with dynamic collections, efficient data manipulation, and practical problem-solving using these fundamental data structures.

Lab Policies

- **Academic Integrity**: Submit your own work. Do not copy code from classmates or online sources. All violations will be reported as academic misconduct.
- **Submission Format**: Submit only the file: **Lab7.java**. Do not upload ZIP files or project folders. Do not change the name of the file as this will not allow us to test your code.
- **Deadline:** Submit your .java file to the eClass course page by **Tuesday, December2 (11:59pm).** No late submissions are accepted. Email submissions are not accepted.
- Your lab assignment is not graded during the weekly lab sessions scheduled.

Learning Outcomes

By the end of this lab, you will be able to:

- Import a starter project archive file.
- Utilize ArrayList methods for dynamic data manipulation including add, remove, get, and size operations.
- Implement HashMap operations for efficient key-value pair storage and retrieval.
- Apply ArrayList for solving problems involving element comparison and collection intersection.
- Use HashMap for frequency counting, grouping, and data categorization tasks.
- Handle edge cases including empty collections, null values, and boundary conditions.
- Understand when to use ArrayList versus HashMap based on problem requirements.
- Use the given JUnit tests to guide the development and verify correctness.

- Use the given JUnit tests (calling the utility methods) to guide the development.

Lab Requirements

- For the JUnit test class Lab7Test.java given to you:
  - Do not modify the test methods given to you.
  - You are allowed to add new test methods.
- For each method in the Lab7 class that you are assigned to implement:
  - Javadoc has been generated for you. You can check the documentation of the project in your browser by clicking on index.html file.
  - No System.out.println statements should appear in each of the utility method.
  - No Scanner operations (e.g., input.nextInt()) should appear in each of the utility methods. Instead, refer to the input parameters of the method.
  - No main method.
  - NO RECURSION - No recursive method calls
  - Don't change the method signatures provided to you. Otherwise, your test cases will fail.
  - Complete the partial code in Lab7.java file to achieve desired solution for the listed programming tasks.
  - A tester file Lab7Test.java is provided to you with some of the test cases implemented.

Download and Import the Starter Project

- Download the Eclipse Java project archive file from eClass: EECS1022_Lab7.zip
- Launch Eclipse and browse to EECS1022-workspace (for instance or your own created workspace).
- In Eclipse:
  - Choose File->Import
  - Under General, choose Existing Projects into workspace
  - Choose Select archive file. Browse your compressed zip folder and attach it.
  - Make sure that the EECS1022_Lab7 box is checked under Projects and you don't have the same project already in the workspace. Then Finish.

EECS1022 – Lab 7: ArrayList and HashMap

Fall 2025 **Due Date: Tuesday, December 2(11:59pm)**

Programming Tasks

You are required to implement the following three methods in the Lab7.java class

Method 1: findCommonElements()

Write a method named findCommonElements that takes two ArrayLists of integers as parameters. The method should return a new ArrayList containing only the elements that appear in both input ArrayLists. Each common element should appear only once in the result, even if it appears multiple times in either input list. The returned ArrayList should maintain the order of elements as they first appear in list1.

Method Signature: `public static ArrayList<Integer> findCommonElements(ArrayList<Integer> list1, ArrayList<Integer> list2)`

Examples:

- `list1 = [1, 2, 3, 4, 5], list2 = [3, 4, 5, 6, 7] → returns [3, 4, 5]`
- `list1 = [10, 20, 30], list2 = [40, 50, 60] → returns [] (empty list)`
- `list1 = [1, 2, 2, 3], list2 = [2, 2, 3, 3] → returns [2, 3] (no duplicates)`

Method 2: removeDuplicates()

Write a method named removeDuplicates that takes an ArrayList of strings as a parameter. The method should return a new ArrayList containing only unique elements from the input list, preserving the order of their first occurrence. The comparison should be case-sensitive.

Method Signature: `public static ArrayList<String> removeDuplicates(ArrayList<String> list)`

Examples:

- `["apple", "banana", "apple", "orange", "banana"] → returns ["apple", "banana", "orange"]`
- `["Java", "Python", "C++", "Java"] → returns ["Java", "Python", "C++"]`
- `["test", "Test", "TEST"] → returns ["test", "Test", "TEST"] (case-sensitive)`

Method 3: rotateLeft()

Write a method named rotateLeft that takes an ArrayList of integers and a positive integer k as parameters. The method should rotate the elements in the list k positions to the left. The rotation

should modify the original list and also return it. If k is greater than the list size, it should wrap around.

Method Signature: `public static ArrayList<Integer> rotateLeft(ArrayList<Integer> list, int k)`

Examples:

- `list = [1, 2, 3, 4, 5], k = 2 → returns [3, 4, 5, 1, 2]`
- `list = [10, 20, 30], k = 1 → returns [20, 30, 10]`
- `list = [1, 2, 3, 4], k = 5 → returns [2, 3, 4, 1] (k wraps around)`

Method 4: findMostFrequent()

Write a method named findMostFrequent that takes an array of integers as a parameter. The method should use a HashMap to count the frequency of each element and return the element that appears most frequently. If there are multiple elements with the same highest frequency, return the one that appears first in the array. If the array is empty, return 0.

Method Signature: `public static int findMostFrequent(int[] arr)`

Examples:

- `[1, 2, 2, 3, 3, 3, 4] → returns 3 (appears 3 times)`
- `[5, 5, 10, 10, 15] → returns 5 (first occurrence of max frequency)`
- `[7] → returns 7`

Method 5: groupByLength()

Write a method named groupByLength that takes an array of strings as a parameter. The method should use a HashMap to group the strings by their length. The keys should be the lengths (integers), and the values should be ArrayLists containing all strings of that length. Return the HashMap. Empty strings should be grouped under key 0.

Method Signature: `public static HashMap<Integer, ArrayList<String>> groupByLength(String[] words)`

Examples:

- `["cat", "dog", "bird", "fish"] → returns {3=[cat, dog], 4=[bird, fish]}`
- `["a", "is", "the", "and"]→ returns {1=[a], 2=[is], 3=[the, and]}`

EECS1022 – Lab 7: ArrayList and HashMap

Fall 2025 **Due Date: Tuesday, December 2(11:59pm)**

Method 6: areAnagrams()

Write a method named areAnagrams that takes two strings as parameters. The method should use a HashMap to determine if the two strings are anagrams (contain the same characters with the same frequencies). The comparison should be case-insensitive and should ignore spaces. Return true if they are anagrams, false otherwise.

Method Signature: `public static boolean areAnagrams(String str1, String str2)`

Examples:

- `"listen", "silent" → returns true`
- `"Hello", "hello" → returns true (case-insensitive)`
- `"a gentleman", "elegant man" → returns true (ignores spaces)`
- `"hello", "world" → returns false`

Note: You may have to ude getOrDefault() method for frequency counting of a key in a hashmap. Syntax: getOrDefault(key, defaultValue)

Returns:

- The **value** associated with num if the key exists in the map
- The **default value** (0 in this case) if the key doesn't exist

Example:

```
HashMap<Integer, Integer> map = new HashMap<>();
map.put(5, 3);  // Key 5 has value 3
// Key exists:
map.getOrDefault(5, 0);  // Returns 3 (the actual value)
// Key doesn't exist:
map.getOrDefault(10, 0); // Returns 0 (the default value)
```

Submission Instructions

- Submit only one file: **Lab7.java**
- Upload it to Lab 7 assignment on eClass.
- Ensure your file compiles and runs without error.
- Submitting files with compilation errors will receive zero marks. No partial marks.
- Your file will be tested with additional test cases.