**Faculty of Computing**

**SE-314: Software Construction**

**Class: BESE 13AB**

# Lab 09: Abstract Data Type - II

**CLO-03:** Design and develop solutions based on Software Construction principles.  
**CLO-04:** Use modern tools such as Eclipse, NetBeans etc. for software construction.

**Name: Affan Rehman**

**CMS: 374064**

**Date: 18th Nov 2024**

**Time: 10:00 AM** **- 12:50 PM   
 02:00 PM – 04:50 PM**

**Instructor: Dr. Mehvish Rashid  
Lab Engineer: Mr. Aftab Farooq**

**Introduction:**

# Lab 09: ADT- II

Students will have hands-on experience on designing, testing, and implementing abstract data types. Given a set of specifications, you will write unit tests that check for compliance with the specifications, and then implement code that meets the specifications. Use GitHub to collaborate with your group members and mention the work distribution.

**Material:**

<https://ocw.mit.edu/ans7870/6/6.005/s16/psets/ps2/>

Lectures on LMS regarding designing Abstract Data Types   
**Lab Tasks**

Solve problem 4 of problem set 2 listed on the link. **Task1: Test Graph Poet**

## Devise, document, and implement tests for GraphPoet in

**GraphPoetTest.java** .

**Task2: Implement GraphPoet in GraphPoet.java .**

## You must use Graph in the rep of GraphPoet , but the implementation is otherwise entirely up to you.

**Solution:**

|  |
| --- |
| **Output:**  **Test:**    **CODE:**  **GraphPoet.java:** package poet;  import java.io.File;  import java.io.IOException;  import java.nio.file.Files;  import java.util.\*;  public class GraphPoet {  private final Map<String, Map<String, Integer>> graph = new HashMap<>();  public GraphPoet(File corpus) throws IOException {  List<String> words = Files.readAllLines(corpus.toPath());  buildGraph(words);  }  public String poem(String input) {  String[] inputWords = input.split("\\s+");  List<String> poemWords = new ArrayList<>();  for (int i = 0; i < inputWords.length - 1; i++) {  String currentWord = inputWords[i].toLowerCase();  String nextWord = inputWords[i + 1].toLowerCase();  poemWords.add(currentWord);  String bridgeWord = findBridgeWord(currentWord, nextWord);  if (!bridgeWord.isEmpty()) {  poemWords.add(bridgeWord.toLowerCase());  }  }  poemWords.add(inputWords[inputWords.length - 1].toLowerCase());  return String.join(" ", poemWords);  }  private String findBridgeWord(String word1, String word2) {  if (!graph.containsKey(word1) || !graph.containsKey(word2)) {  return "";  }  Map<String, Integer> word1Connections = graph.get(word1);  String bridgeWord = "";  int maxWeight = -1;  for (Map.Entry<String, Integer> entry : word1Connections.entrySet()) {  String target = entry.getKey();  int weight = entry.getValue();  if (graph.containsKey(target) && graph.get(target).containsKey(word2)) {  if (weight > maxWeight) {  maxWeight = weight;  bridgeWord = target;  }  }  }  return bridgeWord;  }  private void buildGraph(List<String> words) {  for (int i = 0; i < words.size() - 1; i++) {  String currentWord = words.get(i).toLowerCase();  String nextWord = words.get(i + 1).toLowerCase();  graph.putIfAbsent(currentWord, new HashMap<>());  Map<String, Integer> connections = graph.get(currentWord);  connections.put(nextWord, connections.getOrDefault(nextWord, 0) + 1);  }  }  }  **GraphPoetTest.java:** package poet;  import static org.junit.Assert.\*;  import org.junit.Test;  import java.io.File;  import java.io.IOException;  import java.nio.file.Files;  public class GraphPoetTest {  @Test(expected = AssertionError.class)  public void testAssertionsEnabled() {  assert false; // make sure assertions are enabled with VM argument: -ea  }  @Test  public void testGraphPoetCreationFromFile() throws IOException {  File tempFile = File.createTempFile("test\_corpus", ".txt");  GraphPoet graphPoet = new GraphPoet(tempFile);  assertNotNull(graphPoet);  tempFile.delete();  }  @Test  public void testPoemGenerationDifferentInput() throws IOException {  // Create a temporary corpus file for testing  File tempFile = File.createTempFile("test\_corpus", ".txt");  String corpusContent = "This is a beautiful day in the park.";  Files.write(tempFile.toPath(), corpusContent.getBytes());  // Create GraphPoet instance using the temporary corpus file  GraphPoet graphPoet = new GraphPoet(tempFile);  // Define input and expected poem  String input = "a beautiful day";  String expectedPoem = "a beautiful day";  // Generate poem and check if it matches the expected output  String poem = graphPoet.poem(input);  assertEquals(expectedPoem, poem);  // Delete the temporary file after the test  tempFile.delete();  }  @Test  public void testEmptyInputPoemGeneration() throws IOException {  File tempFile = File.createTempFile("test\_corpus", ".txt");  GraphPoet graphPoet = new GraphPoet(tempFile);  String input = "";  String expectedPoem = "";  String poem = graphPoet.poem(input);  assertEquals(expectedPoem, poem);  tempFile.delete();  }  @Test(expected = IOException.class)  public void testInvalidFile() throws IOException {  File nonExistentFile = new File("nonexistent.txt");  GraphPoet graphPoet = new GraphPoet(nonExistentFile);  }  } |

**- Push Your Code on GitHub  
- Add Git Link in Document.**

**Source Code: Zip your source code and upload one file (Including Git link) on LMS as well.**

**Solution**

**Deliverables:**

Compile a single word document by filling in the solution part and submit this Word file on LMS. In case of any problems with submissions on LMS, submit your Lab assignments by emailing it to [aftab.farooq@seecs.edu.pk.](mailto:aftab.farooq@seecs.edu.pk.)