1: The First Problem

Reinforce your ability to debug and fix errors efficiently. The class DNASequence has methods to examine a DNA sequence. DNA is a polymer composed of four molecules called nucleotides: adenine (A), cytosine (C), guanine (G) and thymine (T). In a computer, a DNA sequence can be represented as a sequence of the characters 'A', 'C', 'G' and 'T': for example, "TTACGGGAGGACGGGGAAAATTACTACGGCATTAGC". The class DNASequence stores the sequence of characters in an object of class String. However, the implementation of class DNASequence has four errors. Use a debugger to help you to debug the program and fix the errors.

2: The Second Problem

This assignment asks you to implement two methods: one that reads employee data from a file and another that writes employee data to a file. The employee data contains basic information (ID, name, and salary) for a collection of employees. In this assignment, you will finish the implementation of EmployeeFileIO. We will provide a test driver and the class Employee.

- Class Employee: A complete implementation of this class is included in the student archive file. You can review its documentation (Employee.html)
- Class EmployeeFileIO: A partial implementation of this class is included in the student archive file. You should complete the implementation of the "read" and "write" methods.
- Class TestEmployeeFileIO: This class is a test driver for EmployeeFileIO. It contains test cases for each method in EmployeeFileIO. A complete implementation is included in the student archive file. You should use this class to test your implementation of EmployeeFileIO. Your implementation of method read is tested by comparing the ArrayList returned by your implementation against an ArrayList returned by our implementation. In the same way, your implementation of method write is tested comparing the file produced by your implementation against a file produced by our implementation.
- Class TestHelper: This class contains auxiliary methods used by the test driver: a method for comparing two ArrayList objects and a method for comparing two files. A complete and compiled implementation is included in the student archive file. Review its documentation and become familiar with it: TestHelper.html. Documentation for class TestHelper

Tasks

Implement the methods read and write in class EmployeeFileIO. Document your code using Javadoc and follow Sun's code conventions. The following steps will guide you through this assignment. Work incrementally and test each increment. Save often.

1. Implement the method read: It begins by creating an empty ArrayList and a BufferedReader object to read data from the specified file. It then proceeds to read each line in the file. After it reads a line, it extracts the ID, name, and salary of an employee, creates an Employee object, and adds the new object to the end of the ArrayList. When all data has been read, it returns the ArrayList. Use BufferedReader.readLine to read the

datainthefile. Use java.util.StringTokenizer to parse the data. You can assume that every line in the file contains the data for exactly one employee in the following format: ID_name_salary. The fields are delimited by an underscore (_). You can assume that the fields themselves do not contain any underscores. The method read should not contain try-catch blocks for the following exceptions. This requirement will simplify the code.

2. Implement the method write: It first creates a PrintWriter object for writing data to the specified file (if the file does not exist, one will be created). It then writes the ID, name, and salary of each employee in the specified ArrayList to the specified file. Every line in the file should contain the data for exactly one employee in the following format: ID_name_salary. The fields are delimited by an underscore (_). The order of the employees in the file should match the order of the employees in the ArrayList. If the specified file exists, its contents should be erased when it is opened for writing. The method write should not contain a try-catch block for the following exception. This requirement will simplify the code.

Upon completion, submit only the EmployeeFileIO.java