National University of Computer and Emerging Sciences

Software Construction and Development

Lab Manual 3

Muhammad Hassan Raza

Fall 2024

Department of Software Engineering
FAST-NU, Lahore, Pakistan

Problem 1: Design a custom generic data structure named `LinkedDeque<T>` that combines the properties of both a Stack and a Queue, supporting operations from both ends. The `LinkedDeque<T>` should be implemented using a doubly linked list and should allow forward and backward iteration using iterators.

1) Implement a generic class `LinkedDeque<T>` with the following operations:
   a) `void addFirst(T element)`: Adds an element to the front of the deque.
   b) `void addLast(T element)`: Adds an element to the end of the deque.
   c) `T removeFirst()`: Removes and returns the element from the front of the deque.
   d) `T removeLast()`: Removes and returns the element from the end of the deque.
   e) `T peekFirst()`: Returns the element at the front of the deque without removing it.
   f) `T peekLast()`: Returns the element at the end of the deque without removing it.
   g) `boolean isEmpty()`: Checks if the deque is empty.
   h) `int size()`: Returns the number of elements in the deque.

2) Implement an inner class `LinkedDequeIterator` that implements the `Iterator<T>` interface. It should allow forward and backward iteration over the elements in the deque.

3) Implement another inner class `ReverseLinkedDequeIterator` that iterates over the deque in reverse order.

4) Create a `main` method where you:
   a) Instantiate `LinkedDeque<Integer>` and `LinkedDeque<String>`.
   b) Add several elements to the deque using `addFirst()` and `addLast()` methods.
   c) Display all elements in both forward and reverse order using iterators.
   d) Remove elements from both ends and display the updated deque.
   e) Display the size of the deque at various stages.

Example Output:

Deque after adding elements:
Front to Back: 5, 10, 15, 20
Back to Front: 20, 15, 10, 5

Removing first and last elements:
Deque after removals:
Front to Back: 10, 15
Size of the deque: 2

Problem 2: Write a program that reads a text file containing student data, processes the data to calculate average marks, the highest and lowest marks, and determines the grade for each student. The processed information should then be written to a new output file.

1) Create a class named `StudentDataProcessor` with the following methods:
   a) `List<Student> readStudentData(String inputFilePath)`: Reads a text file containing student data, creates `Student` objects, and returns a list of students.
   b) `void writeProcessedData(String outputFilePath, List<Student> students)`: Writes the processed data, including student names, average marks, highest mark, lowest mark, and grade to an output file.

2) Create a class named `Student` with the following properties:
   a) `String name`
   b) `List<Integer> marks`

3) The class should have the following methods:
   a) `double calculateAverageMarks()`: Returns the average of marks for that student.
   b) `int getHighestMark()`: Returns the highest mark.
   c) `int getLowestMark()`: Returns the lowest mark.
   d) `String calculateGrade()`: Determines the grade based on average marks (e.g., A, B, C, D, F).

4) The input file should have the following format:
   a) John, 85, 78, 92, 88
   b) Alice, 90, 70, 80, 85
   c) Bob, 75, 82, 78, 80

5) The output file should have the following format after processing:
   a) John: Average Marks = 85.75, Highest Mark = 92, Lowest Mark = 78, Grade = A+
   b) Alice: Average Marks = 81.25, Highest Mark = 90, Lowest Mark = 70, Grade = A-
   c) Bob: Average Marks = 78.75, Highest Mark = 82, Lowest Mark = 75, Grade = B+

6) Implement a `main` method where you:
   a) Read the student data from an input text file using `readStudentData()`.
   b) Process the data to calculate the average marks, highest mark, lowest mark, and grade for each student.
   c) Write the processed data to a new output text file using `writeProcessedData()`.

d) Make sure to handle exceptions related to file handling, such as `FileNotFoundException`, `IOException`, etc.

Note: Provide the input text file named `students.txt` in the working directory of your project.