CS401 Lab 7: Binary Search data structure implementation

Overview

- Focus on implementing various data structures and algorithms.

Part 1: Binary Search and Sorting

Objectives

- Implement *Selection Sort* for *Employee* objects
- Implement *Binary Search* for *Employee* objects

Requirements

- Read employee data from *emp.txt* and create an array of *Employee* objects
- Sort the *Employee* objects by ID using Selection Sort
- Implement a Binary Search function to search for employees by ID

Code Structure

```
class Sorting<T extends Comparable<T>> {
   void selectionSort(T[] array, int low, int high) {
      // Implement Selection Sort
   }

   int binarySearch(T[] array, int low, int high, T key) {
      // Implement Binary Search
   }

   public static void main(String[] args) {
      // Implement test code here
   }
}
```

Part 2: Infix to Postfix Conversion and Evaluation

Objectives

- Convert infix expressions to postfix
- Evaluate postfix expressions using a Stack

Requirements

- Use the Stack class developed in the previous lab
- Implement infix to postfix conversion
- Implement postfix expression evaluation
- Test with the following expressions:
    - 1 + 3 * 8

    - 8 - 3 - 4 * 6 + 3
    - 8 - 2 + 8 / 4 + 6 - 1 - 6 / 2

Expected Output
For each expression, print:
- The original infix expression
- The converted postfix expression
- The evaluated result

```
Infix: 1 + 3 * 8
Postfix: 1 3 8 * +
Evaluation: 25
```

Part 3: Palindrome Checker

Objective
Implement a program to check if a given string is a palindrome

Requirements
- Take input from the user
- Check if the input string is a palindrome
- Print the result

Expected Output
```
Enter a string: racecar
"racecar" is a palindrome.

Enter a string: hello
"hello" is not a palindrome.
```

General Requirements
- Implement each part in a separate Java class
- Include comprehensive inline comments
- Create a README file with:
    - Program description
    - Compilation and execution instructions
    - Brief explanation of implemented algorithms
    - Command to run the JAR file

Submission Requirements
- Source Code Files:
    - Sorting.java (Part 1)
    - InfixPostfixEvaluator.java (Part 2)
    - PalindromeChecker.java (Part 3)
- Compiled Bytecode (.class files)

- Output PDF file containing:
    - Results from Part 1 (sorted employees and search results)
    - Results from Part 2 (all three expression evaluations)
    - Sample runs of Part 3 (palindrome checker)
- Executable JAR file
- README file
- emp.txt (input file for Part 1)

Important Notes

- Ensure proper error handling in all parts
- Test your programs thoroughly with various inputs
- Pay attention to the efficiency of your implementations, especially for sorting and searching in Part 1