COP3530 – Assignment 4

Objective

Students will be able to develop code involving fundamental data structures, by implementing a solution to a problem that requires the use of hash tables and collision resolution using chaining.

Assignment Problem

A *symbol table* is a data structure used by compilers to store information about identifiers in a program such as variables, methods, and classes. In this exercise you are asked to implement a symbol table that will store the identifiers of a Java program and their associated access modifiers (public, private, protected, default/package-private). The following aspects are to be observed:

- Element information is a pair *(identifier, access)*, where *identifier* is the identifier name and *access* is the corresponding access modifier. Both *identifier* and *access* are strings of characters.
- Elements will be read into your program from a text file with one pair (identifier, access) in each line. For example,

    x private
    y public
    p1 protected
    firstName default
    lastName default
    taxValue public

- *SymbolTable*, your main data structure, will be implemented with a hash table that will store the symbol table elements, i.e. pairs (identifier, access). Hash table size will be 13. The key of an element is the identifier component.
- Collisions will be resolved using the chaining approach: elements with the same hash value are stored in a linked list.
- Hash function to be used is the one implemented in class on exercise *Prog23_01*.
- To implement the linked list, use solution in *Prog13_01* (Canvas Home/Week 3) modified to meet the UML diagram given below.
- Several files accompany this project description:
    - a test file *assignment 4 input.txt*,
    - the implementation of the Main class (tests your code using file *assignment 4 input.txt*),
    - the implementation of all *toString* methods required in the assignment,
    - an example of the program output.
- The output of your program is expected to be similar to the output example provided.

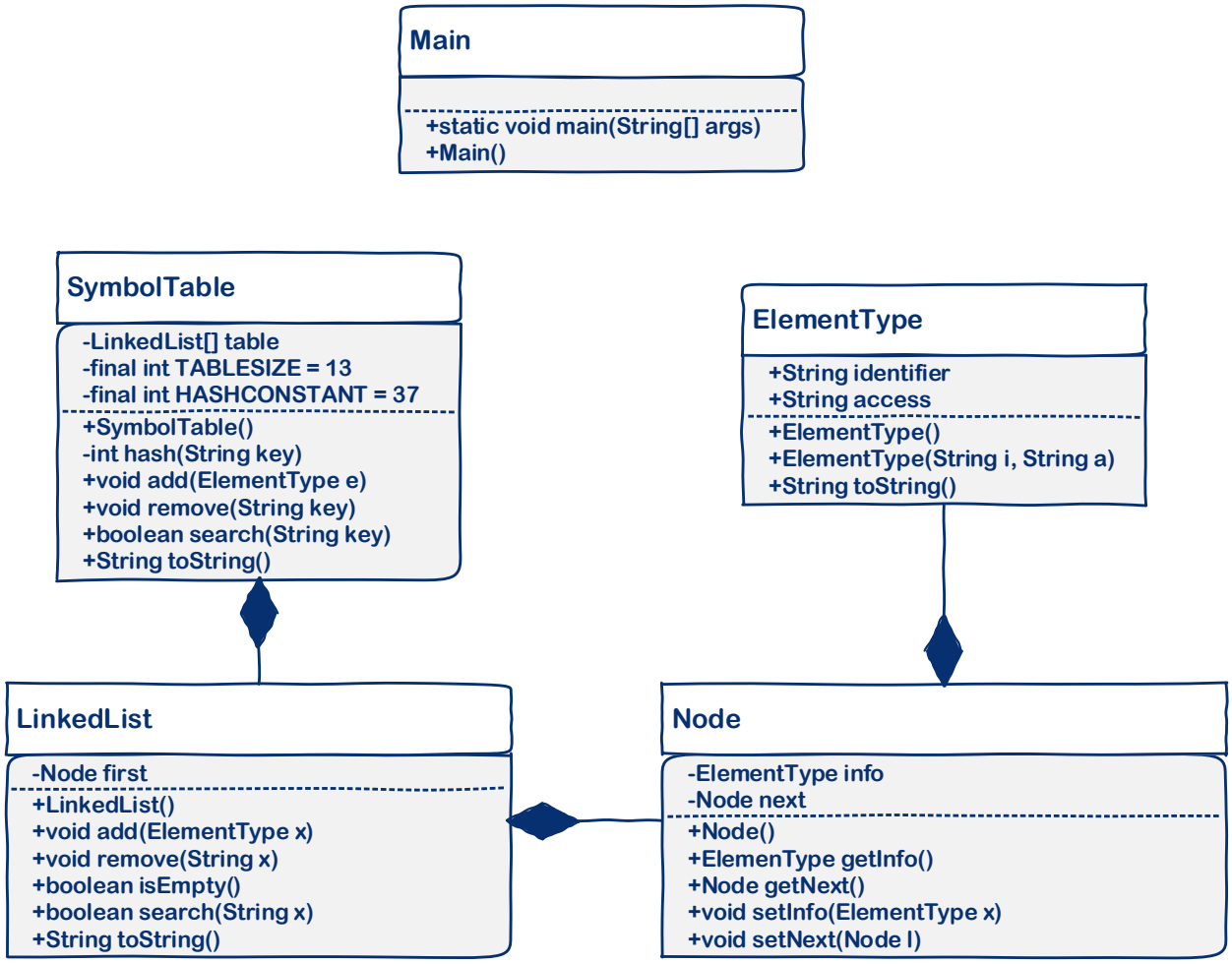
Guidelines

The assignment is to be completed individually. The given problem is based on the content studied on hashing and collision resolution.

You are allowed to use all of the code given in the lectures. In those cases, make sure you properly credit its source.

Design

The design is described in the UML class diagram:



Deliverables:

• A compressed folder, *PID Assignment 4* (e.g. *1234567 Assignment 4*), containing

- all of the source code of the exercise (the .java files; do not include other files or folders generated by the IDE)
- the screenshot of the running program. The screenshot will contain the IDE environment with code (partial view of code is fine) and output window. The output window is expected to be displayed in its entirety.

Make sure you write your Panther ID as the first line of each source file you submit (given as a comment).

Grading Rubric

The assignment is worth 145 points (out of 1000 total course points). Grade components:

<table>
<tr><th>Component</th><th>Points</th><th>Description</th></tr>
<tr><td>Submission</td><td>5</td><td>The student has submitted the project solution using the requirements for deliverables specified in the Deliverables section.</td></tr>
<tr><td>Organization</td><td>5</td><td>Code is expected to be neat, organized, and readable.</td></tr>
<tr><td>Content</td><td>135</td><td>
<table>
<tr><th>Deliverable</th><th>Points</th></tr>
<tr><td>SymbolTable class</td><td>45 pts</td></tr>
<tr><td>LinkedList class</td><td>45 pts</td></tr>
<tr><td>Node class</td><td>22 pts</td></tr>
<tr><td>ElementType class</td><td>14 pts</td></tr>
<tr><td>Screenshot</td><td>9 pts</td></tr>
</table>
</td></tr>
</table>