Program 4 Design

Alexander DuPree

August 6, 2018

1 Task and Purpose

Program number 4's focus is in the design, implementation and manipulation of a binary search tree. This assignment will be nearly identical to program 3, except the Table ADT will utilize the BST as its data structure. As such, the Table ADT will again have the following functionality:

- Add a new item to the table
- Retrieve information on a item by entering the item's key
- Remove an item from the table
- Display all items on the table
- Display all items related to a particular website

2 Design Considerations

Design patterns from object oriented design continue to apply for this assignment. However, utilizing a binary search tree raises new considerations. The performance of the BST is dependent on the balance and shape of the tree. As such, the data for this assignment will be rearranged so that when it is inserted into to the tree we will be able to maintain tree balance. Because the tree is not self-balancing it is important to curate the data set to achieve the best runtime performance.

3 Data Flow

The flow of this assignment is fairly straightforward. On application startup, load in the data set located in 'Shopping.txt' onto the BST. The user will then be able to add/remove/display data based on a search key. The search key will be the name identifier for a given item. Once the user exits the applcation, the program will provide a print out of the BST's efficiency. This printout will include the BST's height, size, and number of leaves.

The following is a summary of the data structures and ADT's directly related to this assignment. Other classes, such as the interface, will be used but has been summarized in other design documents and is a utility classe not directly related to this assignment.

3.1 binary search tree

The binary search tree will be the underlying data structure for the Table ADT. In this program the BST will be specialized to hold c-strings as the key, and item objects as the value. As such the BST will contain the following public interface

Function	Summary	Parameters	Return
insert	Traverses the tree, inserting the data at a leaf	Read only reference to the search key and value objects	A read/write reference to the BST, this allows chaining of methods.
erase	Finds the associated object with a matching key, and removes it from the tree	A valid search key object	True if a item with a matching search key was found and removed
find	Traverses the tree comparing keys, returning a reference to the matched item	A valid search key object	A reference to the object with the matching search key, if found, otherwise a default object reference
clear	Conducts postorder traversal to delete each node	None	None
size	Returns the number of elements in the BST	None	The BST keeps an internal counter of its size so this is a constant time operation
height	Calculates the height of the tree	None	Returns the height of the tree
leaves	Calculates the number of leaves	None	Returns the number of leaves
empty	Tests if the BST is empty	None	returns true if the BST is empty
begin	Returns an iterator to the smallest element in the tree	None	If the BST is empty the iterator will be NULL, this can be checked with the iterators null() method
end	Returns an iterator to one past the largest element of the tree	None	This will be a NULL iterator

Table 1: binary search tree public interface.

3.2 item table

The item table ADT utilizes a binary search tree to manage a table of item objects. The table will utilize the item's name identifier as the search key. The client will be able to interact with the item table with the following public interface:

Function	Summary	Parameters	Return
add item	Adds an item object to the table	A Read only reference to the copyable the item object	True if the operation was successful
remove item	Removes the item with the matching name	A string with the name of the item	True if the operation was successful
get item	Returns the item that matches the search key	A string with the name of the item	reference to the item if found, otherwise a reference to a default item
display all	Uses the BST's iterator methods to display each item in sorted order	None	None
empty	Tests if the table is empty	None	Returns true if the table is empty
size	Returns the number of elements in the table	None	Utilizes the internal counter of the BST so this is a constant time operation

Table 2: item table public interface.

3.3 Item

The Item ADT is a utility class that stores information related to a specific commerical Item. Each Item will contain the following data:

1. Item name
2. Description
3. Color
4. Website to purchase the item

The Item ADT's public interface is fairly straightforward. The Item will contain methods for inspection and mutation of each data member and have the '«' overloaded for displaying the item in a formatted fashion.

A file, 'Shopping.txt' will contain a large data set of these Items attributes. On program startup, 'Shopping.txt' will be parsed, composing each attribute into an Item object and adding that object onto the BST.