Software Design Document for Employee Record, Store, and Customer Database

Programming Assignment 3
March 16, 2020

Prepared By
Paul Abers
pa0034@uah.edu

Prepared For
Mr. James Williamson
CS 221, Data Structures in C++
Computer Science Department
University of Alabama in Huntsville

Contents

1 System Overview

The purpose of this assignment is to provide a simple and easy way to access an employee record database. The database has four parts, the employee record class, a store class, a customer list class and an employee record database. The employee record class must store an employee ID, employee name, department, annual salary of each employee and a customer list class. The customer list class stores a list of store classes. The store class stores information on a single customer store. The employee database will use a binary tree to store employee records.

2 Referenced Documents

Programming Assignment 1 Statement of Work.
Programming Assignment 2 Statement of Work.
Programming Assignment 3 Statement of Work.

3 Architectural Design

3.1 Concept of Execution

This program creates a class structure to store information for an individual employee, a store, a list of stores and a binary search tree of employee records. The employee class stores an employee's first and last name, a unique employee ID, the employee's department ID, the employee's salary, a pointer to a left and right child, and a list of customer stores for the employee. The customer list class stores a list of customer stores.

A database manager will have access to public get and set methods of the class in order to set the various attributes for the employee as well as get them later. There is also a get customer list function. There is also a default constructor that initializes the class as well as a constructor that handles all inputs being included. A quick and easy print will also be provided for quickly displaying all attributes of the class. The customer list class will have functions for adding a store, removing a store, getting a store based on store id and printing a store info. The employee database will have a pointer to the root of the tree and a pointer to an ifstream object.

3.2 Abstract Data Type

The employee record structure is implemented with a class structure separated in a cpp and header file. Likewise for the store class. The customer list is a linked list of store object. The employee database is a binary search tree.

3.3 Code Outline

This program will consist of the following files: EmployeeRecord.h, EmployeeRecord.cpp, Store.h, Store.cpp, CustomerList.h, CustomerList.cpp, EmployeeDatabase.h and EmployeeDatabase.cpp.

EmployeeRecord Class Private Attributes:

- m_iEmployeeID – int value for employee id
- m_sLastName – character array of length 32 for last name
- m_sFirstName – character array of length 32 for first name
- m_iDeptId – int for department id
- m_dSalary – double for employee's salary
- m_pCustomerList – pointer to customer list object
- m_pLeft – pointer to left child
- m_pRight – pointer to right child

Public Methods:

- EmployeeRecord() – default constructor
- EmplyoeeRecord() – initialization constructor
- getID() – return int value of employee id
- setID() – set employee id
- getName() – copy employee's first and last name into pointers passed

- setName() – set employee's first and last name to pointers passed
- getDept() – get value of employee's department
- setDept() – set value of employee's department
- getSalary() – pointer function to get employee's salary
- setSalary() – set employee's salary
- printRecord() – prints to screen all data for employee's record
- getCustomerList() – return the pointer to the employee record's customer list object
- removeCustomerList() – set pointer to customer list to Null
- destroyCustomerList() – deletes the customer list objects and sets pointer to null

Store Class Private Attributes:

- m_iStoreID – pointer integer for store ID
- m_sStoreName – pointer to character array of size 64 for store name
- m_sAdress – pointer to character array of size 64 for address
- m_sCity – pointer to character array of size 32 for city
- m_sState – pointer to character array of size 32 for state
- m_sZip – pointer to character array of size 11 for zipcode

Public Methods:

- m_pNext – pointer to next store for CustomerList object
- Store() – constructor for Store object
- Store() – destructor for Store object

- getStoreID() – return integer for store ID
- setStoreID() – set integer for Store ID
- getStoreName() – get character array for store name
- setStoreName() – set character array for store name
- getStoreAddress() – get character array for store address
- setStoreAddress() – set character array for store address
- getStoreCity() – get character array for store city
- setStoreCity() – set character array for store city
- getStoreState() – get character array for store state
- setStoreState() – set character array for store state
- getStoreZip() – get character array for store zipcode
- setStoreZip() – set character array for store zipcode
- printStoreInfo() – print all the stores info

CustomerList Class <u>Private Attributes</u>:

- m_pHead – pointer to first store in list

<u>Public Methods</u>:

- CustomerList() – constructor for store
- CustomerList() – destructor for store
- addStore() – add a store to list
- removeStore() – remove a store from list
- getStore() – access a store from list

- updateStore() – update a store in list
- printStoresInfo() – print info for each store in list

EmployeeDatabase Class Private Attributes:

- m_pRoot – pointer to root employee record object
- inFile – ifstream object

Public Methods:

- EmployeeDatabase() – constructor for employee database
- EmployeeDatabase() – destructor for employee database
- addEmployee() – add a employee to tree
- removeEmployee() – remove an employee from tree
- getEmployee() – access an employee from tree
- printEmployeeRecords – print employee records of tree
- buildDatabase – build database from an input file

4 Detailed Design

4.1 Source File: EmployeeRecord.h and EmployeeRecord.cpp

4.1.1 Function: EmployeeRecord()

4.1.1.1 Purpose

This is the default constructor for the EmployeeRecord class.

4.1.1.2 Arguments

This default constructor takes no arguments.

4.1.1.3 Return Value

A constructor, therefore no value is returned.

4.1.1.4 Function Outline in Pseudocode

```
Set employee id to 0
set last name to ""
set first name to ""
set department id to 0
set salary to 0.0
set left child to null
set right child to null
```

4.1.1.5 Tracability

This function will fulfil requirement 2.2.2.1 of SOW

4.1.2 Function: EmplyoeeRecord()

4.1.2.1 Purpose

This is the optional constructor to set all values passed into function.

4.1.2.2 Arguments

int employee id, character array pointer first name, character array pointer last name, int for department id, double for salary.

4.1.2.3 Return Value

None

4.1.2.4 Function Outline in Pseudocode

```
Set employee id to ID
copy passed character array for lName into m_sLastName
copy passed character array for fName into m_sFirstName
set department id to dept
set salary to sal.
set left child to null
set right child to null
```

4.1.2.5 Tracability

This function will fulfil requirement 2.2.2.2 of SOW

4.1.3 Function: ~EmplyoeeRecord()

4.1.3.1 Purpose

This is the destructor for the employee record.

4.1.3.2 Arguments

None

4.1.3.3 Return Value

None

4.1.3.4 Function Outline in Pseudocode

```
Properly destruct the class. Clean up and deallocate memory initialized for pointe
character arrays. Properly destruct customer list.
```

4.1.3.5 Tracability

This function will fulfil requirement 2.2.2.3 of SOW

4.1.4 Function: getID()

4.1.4.1 Purpose

This function allows a user to get the private employee ID.

4.1.4.2 Arguments

None

4.1.4.3 Return Value

Int value stored for employee id.

4.1.4.4 Function Outline in Pseudocode

```
Return value of member stored employee ID.
```

4.1.4.5 Tracability

This function will partially fulfil requirement 2.2.2.4 of SOW

4.1.5 Function: setID()

4.1.5.1 Purpose

This function allows a user to set the private employee ID.

4.1.5.2 Arguments

Int value to set the member stored employee id to.

4.1.5.3 Return Value

Void

4.1.5.4 Function Outline in Pseudocode

```
Set internal member variable for employee id to passed integer value.
return
```

4.1.5.5 Tracability

This function will partially fulfil requirement 2.2.2.4 of SOW

4.1.6 Function: getName()

4.1.6.1 Purpose

This function allows a user to get the private employee first and last names.

4.1.6.2 Arguments

Pointer to character array first name, pointer to character array last name

4.1.6.3 Return Value

void

4.1.6.4 Function Outline in Pseudocode

```
Copy contents of internal member variable character arrays for first and last name
character arrays passed into the function.
return
```

4.1.6.5 Tracability

This function will partially fulfil requirement 2.2.2.5 of SOW

4.1.7 Function: setName()

4.1.7.1 Purpose

This function allows a user to set the private employee first name and last name character arrays.

4.1.7.2 Arguments

Pointer to character array first name, pointer to character array last name

4.1.7.3 Return Value

void

4.1.7.4 Function Outline in Pseudocode

```
copy passed character array for lName into m_sLastName
copy passed character array for fName into m_sFirstName
return
```

4.1.7.5 Tracability

This function will partially fulfil requirement 2.2.2.5 of SOW

4.1.8 Function: getDept()

4.1.8.1 Purpose

This function allows a user to get the internal member value for department id.

4.1.8.2 Arguments

noneint reference variable

4.1.8.3 Return Value

void

4.1.8.4 Function Outline in Pseudocode

```
return m_iDeptID
```

4.1.8.5 Tracability

This function will partially fulfil requirement 2.2.2.6 of SOW and 2.0.2.4 of SOW2

4.1.9 Function: setDept()

4.1.9.1 Purpose

This function allows a user to set the employee department id.

4.1.9.2 Arguments

Int for department id

4.1.9.3 Return Value

void

4.1.9.4 Function Outline in Pseudocode

```
Set internal value for department id equal to the passed int value.
```

4.1.9.5 Tracability

This function will partially fulfil requirement 2.2.2.6 of SOW

4.1.10 Function: getSalary()

4.1.10.1 Purpose

Get the employee's salary.

4.1.10.2 Arguments

none

4.1.10.3 Return Value

void

4.1.10.4 Function Outline in Pseudocode

```
return m_dSalary
```

4.1.10.5 Tracability

This function will partially fulfil requirement 2.2.2.7 of SOW

4.1.11 Function: setSalary()

4.1.11.1 Purpose

This function allows a user to set the employee's salary.

4.1.11.2 Arguments

double for salary

4.1.11.3 Return Value

void

4.1.11.4 Function Outline in Pseudocode

```
Set member variable for salary equal to the passed double variable.
return
```

4.1.11.5 Tracability

This function will partially fulfil requirement 2.2.2.7 of SOW

4.1.12 Function: printRecord()

4.1.12.1 Purpose

This function prints all info for employee record to the screen.

4.1.12.2 Arguments

None

4.1.12.3 Return Value

void

4.1.12.4 Function Outline in Pseudocode

```
Print header for employee record
print employee id
print employee first name
print employee last name
print employee department id
print employee salary
return
```

4.1.12.5 Tracability

This function will fulfil requirement 2.2.2.8 of SOW

4.1.13 Function: removeCustomerList()

4.1.13.1 Purpose

This function removes the customer list from employee record and sets pointer to null

4.1.13.2 Arguments

None

4.1.13.3 Return Value

void

4.1.13.4 Function Outline in Pseudocode

```
check if pointer to customer is null
if not null
    set pointer to customer list to null
return
```

4.1.13.5 Tracability

This function will fulfil requirement 2.2.2 of SOW 3

4.1.14 Function: destroyCustomerList()

4.1.14.1 Purpose

This function deletes the customer list from employee record and sets pointer to null

4.1.14.2 Arguments

None

4.1.14.3 Return Value

void

4.1.14.4 Function Outline in Pseudocode

```
check if pointer to customer is null
if not null
    destroy the customer object
    set pointer to customer list to null
return
```

4.1.14.5 Tracability

This function will fulfil requirement 2.2.2 of SOW 3

4.2 Source File: CustomerList.h and CustomerList.cpp

4.2.1 Function: CustomerList()

4.2.1.1 Purpose

This is the default constructor for the CustomerList class.

4.2.1.2 Arguments

This default constructor takes no arguments.

4.2.1.3 Return Value

A constructor, therefore no value is returned.

4.2.1.4 Function Outline in Pseudocode

```
set m_pHead to Null
```

4.2.1.5 Tracability

This function will fulfil requirement 2.2.2.1 of SOW by providing a default constructor for the employee record class.

4.2.2 Function: ˜CustomerList()

4.2.2.1 Purpose

This is the default destructor for the CustomerList class.

4.2.2.2 Arguments

This default constructor takes no arguments.

4.2.2.3 Return Value

A constructor, therefore no value is returned.

4.2.2.4 Function Outline in Pseudocode

```
set curr equal to m_pHead
iterate through list until curr's next store id is null
    set next equal to curr next pointer
    delete curr
    set curr equal to next
delete curr
return
```

Start at m_pHead
Loop over each Store object in linked list
delete each Store object

4.2.2.5 Tracability

This function was not specified in Statement of Work.

4.2.3 Function: addStore()

4.2.3.1 Purpose

Add a store to the customer list.

4.2.3.2 Arguments

A pointer to a store object.

4.2.3.3 Return Value

Bool to indicate success of insertion.

4.2.3.4 Function Outline in Pseudocode

```
set success equal to false
set curr equal to m_pHead
Iterate through list until curr’s next store id is null
    if curr next store id is greater than input store id
        if curr id is less than input store id
            set input store next pointer equal to curr next pointer
            set curr next pointer equal to input store
            set success equal to true
            break loop
return success
```

Set value for m_pNext of Store object equal to store object pointer argument
return true

4.2.3.5 Tracability

This function will fulfil requirement 2.0.4.2.1 of SOW 2.

4.2.4 Function: removeStore()

4.2.4.1 Purpose

Remove a store from the customer list.

4.2.4.2 Arguments

Integer for store with id ID to remove

4.2.4.3 Return Value

A pointer to a store object.

4.2.4.4 Function Outline in Pseudocode

```
set temp equal to null
set curr equal to m_pHead
Iterate through list until curr’s next store equals null
    if curr’s next store id equals input id
```

```
        set temp equal to curr next store
        set curr next store equal to temp next store
        set temp next store equal to null
        break loop
return temp
```

Start at m_pHead
Iterate through list until m_pHead's next store's id equals input id.
Set temporary store variable equal to m_pHead's next store
Set m_pHead's next pointer equal to m_pHead's next next pointer
Set temporary store variables next pointer equal to NULL
break loop iteration
return temporary store.

4.2.4.5 Tracability

This function will fulfil requirement 2.0.4.2.2 of SOW2

4.2.5 Function: getStore()

4.2.5.1 Purpose

Return a pointer to a store object with a given Id if in the list.

4.2.5.2 Arguments

Integer for store ID to get.

4.2.5.3 Return Value

a pointer to a store object if Id found, else NULL

4.2.5.4 Function Outline in Pseudocode

```
set curr equal to m_pHead
Iterate through list until curr's next store equals null
    if curr id equals input id
        break loop
return curr
```

4.2.5.5 Tracability

This function will fulfil requirement 2.0.4.2.3 of SOW2

4.2.6 Function: updateStore()

4.2.6.1 Purpose

Update a stores value

4.2.6.2 Arguments

Integer for store Id to update, char array for name of store, char array for address of store, char array for city of store, char aray for street of store, char array for zipcode of store.

4.2.6.3 Return Value

A boolean to indicate success of insertion.

4.2.6.4 Function Outline in Pseudocode

```
Set success equal to false
Set found equal to false
Set curr equal to m_pHead
Iterate through list until curr's next store equals input id
    Set found equal to true
    break loop
If found is true
    advance curr to next store
    update all the store's value
    set success equal to true
return success
```

set success equal to false
Start at m_pHead
Iterate through list until m_pHead's next store's id equals input id.
If m_pHead's next store id equals input id, call all set functions for next store data with input args
set success equal to true

break loop
return success

4.2.6.5 Tracability

This function will fulfil requirement 2.0.4.2.4 of SOW2

4.2.7 Function: printStoresInfo()

4.2.7.1 Purpose

Print all store info for all stores in customer list

4.2.7.2 Arguments

No arguments

4.2.7.3 Return Value

void

4.2.7.4 Function Outline in Pseudocode

```
Start loop at m_pHead
Loop over each store in customer list
    print store info
return
```

4.2.7.5 Tracability

This function will fulfil requirement 2.0.4.2.5 of SOW2

4.3 Source File: EmployeeDatabase.h and EmployeeDatabase.cpp

4.3.1 Function: EmployeeDatabase()

4.3.1.1 Purpose

This is the default constructor for the EmployeeDatabase class.

4.3.1.2 Arguments

This default constructor takes no arguments.

4.3.1.3 Return Value

A constructor, therefore no value is returned.

4.3.1.4 Function Outline in Pseudocode

```
Set m_pRoot to NULL
set inFile to NULL
```

4.3.1.5 Tracability

This function will fulfil requirement 2.3.2.1 of SOW 3

4.3.2 Function: ~EmployeeDatabase()

4.3.2.1 Purpose

This is the default destructor for the EmployeeDatabase class.

4.3.2.2 Arguments

This default destructor takes no arguments.

4.3.2.3 Return Value

A destructor, therefore no value is returned.

4.3.2.4 Function Outline in Pseudocode

```
If root is not null
    next left is root left child
    next right is root right child
    delete(next left)
    delete(next right)
delete(root)
return
```

4.3.2.5 Tracability

This function will fulfil requirement 2.3.2.1 of SOW 3

4.3.3 Function: addEmployee()

4.3.3.1 Purpose

Add an employee to the EmployeeDatabase tree.

4.3.3.2 Arguments

pointer to an EmployeeRecord object

4.3.3.3 Return Value

bool for success of insertion

4.3.3.4 Function Outline in Pseudocode

```
if root is null
    root equals employee record
else
    addEmployeeHelp(root, employeerecord)
return true
```

4.3.3.5 Tracability

This function will partially fulfil requirement 2.3.2.2 of SOW 3

4.3.4 Function: addEmployeeHelp()

4.3.4.1 Purpose

Add an employee to the EmployeeDatabase tree.

4.3.4.2 Arguments

pointer to current node, pointer to an EmployeeRecord object

4.3.4.3 Return Value

void

4.3.4.4 Function Outline in Pseudocode

```
if node employee record is greater than employee record id
    if node left child is null
        node left child equals employee record
    else
        addEmployeeHelp(node left child, employee record)
elif node employee record is less than employee record id
    if node right child is null
        node right child equals employee record
    else
        addEmployeeHelp(node right child, employee record)
else
    return false
return true
```

4.3.4.5 Tracability

This function will partially fulfil requirement 2.3.2.2 of SOW 3

4.3.5 Function: getEmployee()

4.3.5.1 Purpose

get an employee from the EmployeeDatabase tree.

4.3.5.2 Arguments

integer id of employee to get

4.3.5.3 Return Value

pointer to employee record object found

4.3.5.4 Function Outline in Pseudocode

```
return getEmployeeHelp(root, id)
```

4.3.5.5 Tracability

This function will partially fulfil requirement 2.3.2.3 of SOW 3

4.3.6 Function: getEmployeeHelp()

4.3.6.1 Purpose

get an employee from the EmployeeDatabase tree helper function.

4.3.6.2 Arguments

current node to search at, integer id of employee to get

4.3.6.3 Return Value

pointer to employee record object found

4.3.6.4 Function Outline in Pseudocode

```
if node is null
    return null
elif node id equals id
    return node
elif node employee record id is greater than  id
    return getEmployeeHelp(node left child, id)
elif node employee record id is less than id
    return getEmployeeHelp(node right child, id)
else
    return null
```

4.3.6.5 Tracability

This function will partially fulfil requirement 2.3.2.3 of SOW 3

4.3.7 Function: removeEmployee()

4.3.7.1 Purpose

remove an employee from the EmployeeDatabase tree.

4.3.7.2 Arguments

integer id of employee to remove

4.3.7.3 Return Value

pointer to employee record object removed

4.3.7.4 Function Outline in Pseudocode

```
if root is null
    return null
elif root id equals id
    if root left child and root right child are null
        temp equals root
        root equals null
        return temp
    elif root left child is null
        temp equals root
        root equals root right child
        return temp
    elif root right child is null
        temp equals root
        root equals root left child
        return temp
else
    return removeEmployeeHelp(root, id)
```

4.3.7.5 Tracability

This function will partially fulfil requirement 2.3.2.3 of SOW 3

4.3.8 Function: removeEmployeeHelp()

4.3.8.1 Purpose

remove an employee from the EmployeeDatabase helper function.

4.3.8.2 Arguments

node to check at, integer id of employee to remove

4.3.8.3 Return Value

pointer to employee record removed

4.3.8.4 Function Outline in Pseudocode

```
if node id is greater than id
    if node left child id equals id
        return handleRemoveEmployee(node, id)
    elif node left child is null
        return null
    else
        return removeEmployeeHelp(node left child)
if node id is less than id
    if node right child id equals id
        return handleRemoveEmployee(node, id)
    elif node right child is null
        return null
    else
        return removeEmployeeHelp(node left child)
```

4.3.8.5 Tracability

This function will partially fulfil requirement 2.3.2.4 of SOW 3

4.3.9 Function: handleRemoveEmployee()

4.3.9.1 Purpose

remove an employee from the EmployeeDatabase handle function.

4.3.9.2 Arguments

node to remove

4.3.9.3 Return Value

pointer to employee record removed

4.3.9.4 Function Outline in Pseudocode

```
if node left child id equals id
    temp equals node left child
    if temp left child and right child are null
        node left child equals temp left child
    elif temp left child is null
        node left child equals temp right child
    elif temp right child is null
        node left child equals temp left child
    else
        predecessor equals getPredecessor(temp, temp left child)
        node left child equals predecessor
else
    temp equals node right child
    if temp left child and right child are null
        node right child equals temp right child
    elif temp left child is null
        node right child equals temp right child
    elif temp right child is null
        node right child equals temp left child
    else
        predecessor equals getPredecessor(temp, temp left child)
        node right child equals predecessor
return temp
```

4.3.9.5 Tracability

This function will partially fulfil requirement 2.3.2.4 of SOW 3

4.3.10 Function: getPredecessor()

4.3.10.1 Purpose

get predecessor for node to remove

4.3.10.2 Arguments

node to remove, node to remove traverser

4.3.10.3 Return Value

pointer to employee record predecessor

4.3.10.4 Function Outline in Pseudocode

```
if node traverser right child is not null
    return getpredecessor(node, node traverser right child)
else
    if node traverser left child is not null
        node right child equals node traverser left child
    return node traverser
```

4.3.10.5 Tracability

This function will partially fulfil requirement 2.3.2.4 of SOW 3

4.3.11 Function: printEmployeeRecords()

4.3.11.1 Purpose

print employee records database public function

4.3.11.2 Arguments

no arguments

4.3.11.3 Return Value

void

4.3.11.4 Function Outline in Pseudocode

```
    printEmployeeRecords(root)
```

4.3.11.5 Tracability

This function will fulfil requirement 2.3.2.5 of SOW 3

4.3.12 Function: printEmployeeRecords()

4.3.12.1 Purpose

print employee records database private function

4.3.12.2 Arguments

node to print

4.3.12.3 Return Value

void

4.3.12.4 Function Outline in Pseudocode

```
if node is null
    return
printEmployeeRecords(node left child)
node.printEmployeeRecord()
printEmployeeRecords(node right child)
return
```

4.3.12.5 Tracability

This function will fulfil requirement 2.3.3.1 of SOW 3

4.3.13 Function: buildDatabase()

4.3.13.1 Purpose

build database

4.3.13.2 Arguments

character array for input data file

4.3.13.3 Return Value

bool for success of build

4.3.13.4 Function Outline in Pseudocode

```
Code is provided
```

4.3.13.5 Tracability

This function will fulfil requirement 2.3.2.6 of SOW 3

4.3.14 Function: destroyTree()

4.3.14.1 Purpose

recursively traverse the tree and delete all nodes

4.3.14.2 Arguments

root of tree

4.3.14.3 Return Value

void

4.3.14.4 Function Outline in Pseudocode

```
if node is null
    return
destroyTree(node left child)
destroyTree(node right child)
delete node
return
```

4.3.14.5 Tracability

This function will partially fulfil requirement 2.3.3.2 of SOW 3