```
// ---------------------------------store.h---------------------------------
// Adam Ali CSS 343 A
// Created: 05/21/17
// Modified: 05/23/17
// -------------------------------------------------------------------------
// Describes the ADT Store such that collections of information + records
// can be maintained to represent a movie store that is setup for media item
// checkouts, much like a library. Any given Store will maintain a record
// of media items, customers, and the transaction history of said customers.
// -------------------------------------------------------------------------
// Functionality includes:
//              - create a no-name Store
//              - create a named Store
//              - copy an existing Store
//              - destruct a Store
//              - build a Customer hash table from file
//              - build a Media tree hash table from file
//              - honor borrow/return/history requests for any Customer
//              - output all Media and Customers of a Store

#pragma once

#include <string>
#include <iostream>
#include "customer.h"
#include "media.h"
#include "bintree.h"
using namespace std;

class Store {
        public:
                // -------------------------------Store------------------------------
                // Store: creates a no-name Store.
                // preconditions: none.
                // postconditions: a no-name Store is created.
                // ------------------------------------------------------------------
                Store();


                // -------------------------------Store------------------------------
                // Store: creates a named Store.
                // preconditions: none.
                // postconditions: a named Store is created.
                // ------------------------------------------------------------------
                Store(std::string);


                // -------------------------------Store------------------------------
                // Store: creates a copy of the other Store.
                // preconditions: none.
                // postconditions: a copy Store is created.
                // ------------------------------------------------------------------
                Store(const Store&);


                // ------------------------------~Store------------------------------
                // ~Store: destructs the Store and frees any assoc. memory.
                // preconditions: none.
                // postconditions: any assoc. memory is freed, object inaccessible.
                // ------------------------------------------------------------------
                ~Store();


                // ---------------------------buildCustomers-------------------------
                // buildCustomers: builds Customer hash table from file.
                // preconditions: file is properly formatted, i.e. each line is
                //                                  #### fname lname
                //                                  given 4 digit IDs, the maximum allowed is 10,000.
                // postconditions: customer hash table populated with all entries.
                // -----------------------------------------------------------------
                bool buildCustomers(istream&) {
                        // while lines are available
                                // read current line
                                // extract id, fname, lname
                                // build a Customer with extracted data
                                // generate hash from id
                                // set members[hash % MAX_CUSTOMERS] = freshly built Customer
                }


                // ---------------------------buildInventory--------------------------
                // buildInventory: builds Media tree from file.
                // preconditions: file is properly formatted, i.e. each line is
                //                                  X, ##, dfname dlname, title, . . .
                //                                  genre, stock, director name, title, and more
attribs
                //                                  comma delimited
                // postconditions: BinTree populated with media items.
                // -----------------------------------------------------------------
                bool buildInventory(istream&) {
                        // while lines are available
                                // read current line
                                // extract genre, stock, director, title
                                // if genre is Drama/Funny
                                        // extract year
                                        // build Drama/Funny
                                // else
                                        // extract actor fname/lname, month, year
                                        // build Classic
                                // hash genre char
                                // inventory[hash % MAX_GENRES].insert(Media)
                }


                // -------------------------executeTransactions---------------------
                // executeTransactions: process transactions from file and update
                //                                                 records as needed.
                // preconditions: file is properly formatted, i.e. each line is one of
                //                                  X #### X X # #### fname lname
                //                                  X #### X X Title, ####
                //                                  X #### X X fname lname, title
                //                                  X ####
                //                                  X
                // postconditions: records updated according to transactions.
                // -----------------------------------------------------------------
                bool executeTransactions(istream&) {
                        // while lines available
                                // read current line
                                // extract trans type
                                // switch ( trans type )

                                        // case I : inventory.display()
                                        // case H : read id
                                                        // displayHistory(id);
                                        // case B : read id, format, genre
                                                        // if Funny, read title, year
                                                        // if Drama, read director, title
                                                        // if Classic, read month, year,
actor f/lname
```

```
                                                        // build Movie item from collected
data
                                                        // inventory.retrieve(Movie, ptr);
                                                        // borrowItem(id, ptr);
                                // case R : read id, format, genre
                                                        // if Funny, read title, year
                                                        // if Drama, read director, title
                                                        // if Classic, read month, year,
actor f/lname
                                                        // build Movie item from collected
data
                                                        // inventory.retrieve(Movie, ptr);
                                                        // returnItem(id, ptr);
                }


                // -----------------------------borrowItem--------------------------
                // borrowItem: applies a borrow transaction for a particular customer
                //                          + media item.
                // preconditions: customer exists, 0 < id < 10,000
                //                             media item exists.
                // postconditions: customer transaction history + item stock updated.
                // ----------------------------------------------------------------
                bool borrowItem(int, Media*) {
                        // build Transaction object ('B', ptr);
                        // if ptr != NULL, get hash for id
                        // members[id].addTransaction(Transaction)
                        // ptr->decreaseCount()
                }


                // -----------------------------returnItem--------------------------
                // returnItem: applies a return transaction for a particular customer
                //                          + media item.
                // preconditions: customer exists, 0 < id < 10,000
                //                             media item exists.
                // postconditions: customer transaction history + item stock updated.
                // ----------------------------------------------------------------
                bool returnItem(int, Media*) {
                        // build Transaction object ('R', ptr);
                        // if ptr != NULL, get hash for id
                        // members[id].addTransaction(Transaction)
                        // ptr->increaseCount()
                }


                // ---------------------------displayHistory------------------------
                // displayHistory: outputs the transaction history for a particular
                //                                  customer.
                // preconditions: customer exists, 0 < id < 10,000
                // postconditions: customer transaction history, if any, is output.
                // ----------------------------------------------------------------
                bool displayHistory(int) const {
                        // get id hash
                        // members[hash].displayHistory();
                }


                // ------------------------------display----------------------------
                // display: displays all customers and media items in the store,
                //                       + name if a named store.
                // preconditions: none.
                // postconditions: all Store info output to console.
                // ----------------------------------------------------------------
                void display() const;
```

```
            // --------------------------displayCustomers-----------------------
            // displayCustomers: outputs all customers.
            // preconditions: none.
            // postconditions: all customers, if any, are output as
            //                                 #### fname lname
            // -----------------------------------------------------------------
            void displayCustomers() const;


            // --------------------------displayInventory-----------------------
            // displayInventory: outputs all media items.
            // preconditions: none.
            // postconditions: all media items, if any, are output as they
            //                                 appeared in input.
            // -----------------------------------------------------------------
            void displayInventory() const;


            // ------------------------------getName----------------------------
            // getName: obtains store name, if any.
            // preconditions: none.
            // postconditions: store name returned.
            // -----------------------------------------------------------------
            string getName() const;


            // ---------------------------customerExists------------------------
            // customerExists: determines whether a customer ID exists in the
            //                                 Store.
            // preconditions: 0 < id < 10,000
            // postconditions: true if id found, otherwise false.
            // -----------------------------------------------------------------
            bool customerExists(int) const;


            // ----------------------------getCustomer--------------------------
            // getCustomer: obtains customer info matching id.
            // preconditions: 0 < id < 10,000
            // postconditions: customer returned, if found.
            // -----------------------------------------------------------------
            Customer getCustomer(int) const;

        private:
            string name; // title of business
            const static int MAX_CUSTOMERS = 10000;
            const static int MAX_GENRES = 53;               // 2x alphabet, closest prime

            Customer members[MAX_CUSTOMERS]; // customer hash table
            BinTree inventory[MAX_GENRES];   // media inventory

            // -----------------------------hash--------------------------------
            // hash: obtains a hash based on customer name, for the Customer hash
            //               table + genre char for the Media tree.
            // preconditions: string is nonempty.
            // postconditions: Customer hash returned.
            // -----------------------------------------------------------------
            int hash(string) const;
    };
```