```
// --------------------------------customer.h--------------------------------
// Adam Ali / Hyosang Park CSS 343 A
// Created: 05/21/17
// Modified: 05/23/17
// --------------------------------------------------------------------------
// Describes the ADT Customer such that a record of id #, first/last name, and
// transaction history regarding media borrows/returns and history output
// requests can be recorded.
//
// This customer class deals with every customer-related information.
// For example, id and name information of each customer are used to identify
// each customer and history vector is used to store each customer's
// transaction history. Therefore, we can know what items each customer
// borrowed or returned.

// --------------------------------------------------------------------------
// Functionality includes:
//              - create a Customer
//      - copy a Customer
//      - destruct a Customer
//      - retrieve id/first/last
//      - display Customer info/history
//      - add a new Transaction to the Customer transaction historu
//      - evaluat whether two Customers are equivalent

#pragma once

#include <string>
#include <vector>
#include "transaction.h"
using namespace std;

class Customer {
        public:
                // -------------------------------Customer-----------------------------
                // Customer: creates a unique Customer.
                // preconditions: 0 <= id < 10,000, strings non-empty
                // postconditions: specified Customer created.
                // --------------------------------------------------------------------
                Customer(int, string, string);


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


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


                // --------------------------------getID-------------------------------
                // getID: obtains unique ID.
                // preconditions: none.
                // postconditions: unique ID returned.
                // --------------------------------------------------------------------
                int getID() const;
```

```
// ------------------------------getFirst-------------------------------
// getFirst: obtains the first name.
// preconditions: none.
// postconditions: first name returned.
// ---------------------------------------------------------------------
string getFirst() const;


// ------------------------------getLast--------------------------------
// getFirst: obtains the last name.
// preconditions: none.
// postconditions: last name returned.
// ---------------------------------------------------------------------
string getLast() const;


// ------------------------------display--------------------------------
// display: outputs unique ID + f/lname
// preconditions: none.
// postconditions: identity output to console.
// ---------------------------------------------------------------------
void display() const;


// -------------------------displayHistory---------------------------
// displayHistory: outputs all Transaction history.
// preconditions: none.
// postconditions: all available Transaction history is output.
// ---------------------------------------------------------------------
void displayHistory() const {
        // for all transactions in vector
                // transaction.display();
}


// -------------------------addTransaction---------------------------
// addTransaction: add a Transaction to history as most recent.
// preconditions: none.
// postconditions: history vector includes Transaction as latest.
// ---------------------------------------------------------------------
bool addTransaction(Transaction) {
        // history.push_back(Transaction);
}


// ----------------------------operator==-------------------------------
// operator==: checks whether two customers are exactly same by using
//             each customer's id number, because every customer gets
//             a unique id number.
// preconditions: none.
// postconditions: true if identical, false otherwise.
// ---------------------------------------------------------------------
bool operator==(const Customer&) const;


// ----------------------------operator!=-------------------------------
// operator!=: checks whether two customers are not the same by using
//             each customer's id number, because every customer gets
//             a unique id number.
// preconditions: none.
// postconditions: true if different, false otherwise.
// ---------------------------------------------------------------------
bool operator!=(const Customer&) const;
```

```
        private:
                int id;                                                 // customer unique id #
                string fname;                                   // first name
                string lname;                                   // last name
                vector<Transaction> history;    // transaction history
};
```