// --------------------------- Movie\_Rental\_Store Design Overview ------------------------------

// Group12: Siqi Zheng, Ruying Chen, Junjie Liu, Yi Li

// Assignment4, CSS502

// Date of Last Modification : 2/25/2020

// Created on 2/23/2020

// -------------------------------------------------------------------------------------------

//Basically, this system incorporates three parts: customer database, movie inventory and store //manager.

//The Store manager is the core hub of this system. Its duty is to read customer info, movie info //and commands info from .txt files. Movie information is stored in three separate Binary search //trees contained in a binary search tree array which is the essence of our inventory //implementation, by making use of movie factory creating different movie objects. Customer //information is stored in hash table, which is the core nature of customer database. Commands //will be executed by making use of transactionFactory by creating different transaction objects //and doing different operations. Then using subscribeTransaction function to make the inventory //changed accordingly. This function is just like an event listener, when certain transaction //happens, changing inventory would be triggered.

//The CustomerDatabase is encapsulated with HashTable data structure of Customer type. Class //Customer has-a Transaction, so that we can easily display each customer’s transaction history. //Three types of Transaction, which are Return, Borrow, History inherits Transaction, along with //transactionFactory associates with them, creating different Transaction objects (Return object, //Borrow object, History object ) according to Store manager’s instruction.

//The Inventory is encapsulated with Binary Search Tree data structure of Movie type. Each tree //node has a Movie class, thus we need implementing comparison of Movie object, as I provide //on the UML. Three types of Movie, which are Classic, Drama, Comedy inherits Movie class, along //with movieFactory associates with them, creating different Movie objects (Classic object, Drama //object, Comedy object ) according to Store manager’s instruction.

// ------------------------------------------------main.cpp--------------------------------

// Purpose: main program logic. Provides an entrance to the program. Initiate Store object, and // executes reading from files

// -------------------------------------------------------------------------------------------

#include <iostream>

#include "store.h"

int main()

{

Store myStore;

myStore.readMovieFile();

myStore.readCustomerFile();

myStore.readCommandsFile();

return 0;

}

// --------------------------- Pseudocode for myStore.readMovieFile()------------------------------

// Purpose: read movie info, executes storing movie information in separate Binary search trees //hosted in a binary search tree array which is the essence of our inventory implementation, by //making use of movie factory creating different movie object.

// -------------------------------------------------------------------------------------------

MovieFactory movieFactory;

Inventory inventory;

while(is not end of file)

{

char c = the first charactor read;

Movie\* m = movieFactory(c, filestream);

m->setData(filestream);

if( inventory.movies[c - 'a'] == NULL)

{

inventory.movies[c - 'a'] = new BinTree;

}

inventory.movies[c - 'a']->insert(m, node);

}