CSSE 343 – FALL 2014

Instructed by: *Dr. Min Chen*

**ASSIGNMENT 4 – DESIGN DOCUMENT**

***Prepared by***: *Nghiep Ngo, Jessica Li, Nikita Borisov, Seth Pham*

Table of Contents

[**I.** **OVERVIEW** 2](#_Toc403742542)

[**II.** **ANALYSIS & DESIGN** 3](#_Toc403742543)

[**1)** **Class Diagram** 3](#_Toc403742544)

[**2)** **Class Description** 4](#_Toc403742545)

[**2.1** **- Store** 4](#_Toc403742546)

[**2.2 - InventoryManager** 5](#_Toc403742547)

[**2.2.1 MovieFactory** 5](#_Toc403742548)

[**2.2.2 Movie** 5](#_Toc403742549)

[**2.2.3 Classic** 6](#_Toc403742550)

[**2.2.4 Comedy** 6](#_Toc403742551)

[**2.2.5 Drama** 6](#_Toc403742552)

[**2.2.6- BSTree** 7](#_Toc403742553)

[**2.3 - CustomerManager** 8](#_Toc403742554)

[**2.3.1 Customer** 9](#_Toc403742555)

[**2.3.2** **HashTable** 10](#_Toc403742556)

[**2.4 - TransactionManager** 11](#_Toc403742557)

[**2.4.1 TransactionFactory** 11](#_Toc403742558)

[**2.4.2 Transaction** 12](#_Toc403742559)

[**2.4.3 Borrow** 12](#_Toc403742560)

[**2.4.4 Return** 13](#_Toc403742561)

[**2.4.5 History** 13](#_Toc403742562)

[**2.4.6 Inventory** 13](#_Toc403742563)

[**3)** **Memory Diagram** 14](#_Toc403742564)

[**3.1 Hash Table** 14](#_Toc403742565)

[**3.2 BSTree** 14](#_Toc403742566)

# **OVERVIEW**

* Our design includes a class Store, which has an InventoryManager, a CustomerMananger, and a TransactionManager.
* An InventoryManager contains a collection of store Items in binary search trees for quick look-up based on different sorting criteria. Item is the base class for different types of inventory items in different media. An ItemFactory class creates new Items using factory method pattern. For this project we would only consider Movie in DVDs. Item and ItemFactory classes are for possible extension only and won't be implemented for this project. InventoryManager provides interfaces to add new Movies delete obsolete Movies, and update stock when an Item is checked out or returned. InventoryManager can also process a formatted file containing inventory information.

Movie is the base class for Comedy, Drama and Classics. MovieFactory creates new Movie using factory method pattern.

* A CustomerManager maintains a collection of Customers implemented by a hash table, identified by their 4-digit Customer ID. CustomerManager adds and deletes Customers. CustomerManager is responsible for maintaining each Customer account.

CustomerManager can also process a formatted file containing customer information.

Customer class contains last name, first name and Customer ID. In addition it maintains a queue of Transaction history and a queue of Movies currently checked out by the Customer.

* A TransactionManager processes Transaction requests. A TransactionFactory makes new Transactions using factory method pattern. A TransactionManager can process a single Transaction or a batch of Transactions such as an input file, by calling the InventoryManager to update stock and calling the CustomerManager to update individual Customer account.

A TransactionManager also maintains a queue for transaction output. Transaction is the base class for Borrow, Return, Inventory, and History.

int Main()

{

Store myStore;

ifstream infile1("data4movies.txt");

ifstream infile2("data4customers.txt");

ifstream infile3("data4commands.txt");

myStore.processInventory(infile1);

myStore.processCustomers(infile2);

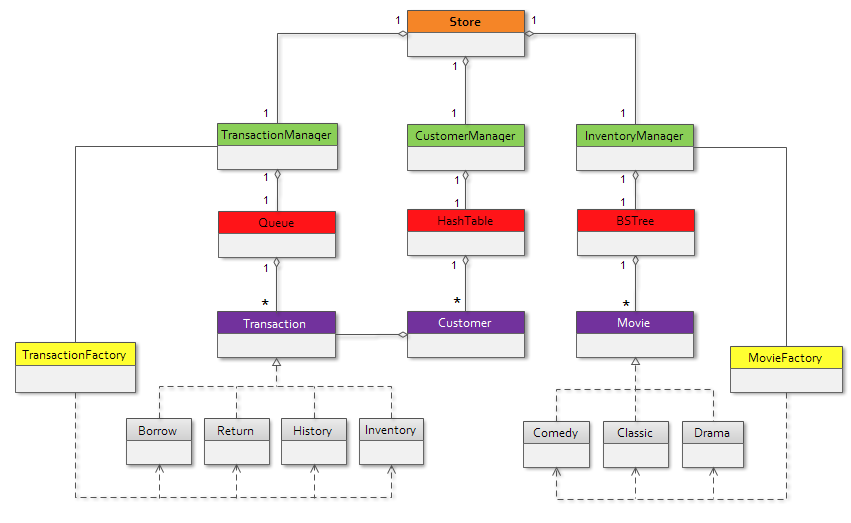
myStore.processTransactions(infile3);

}

# **ANALYSIS & DESIGN**

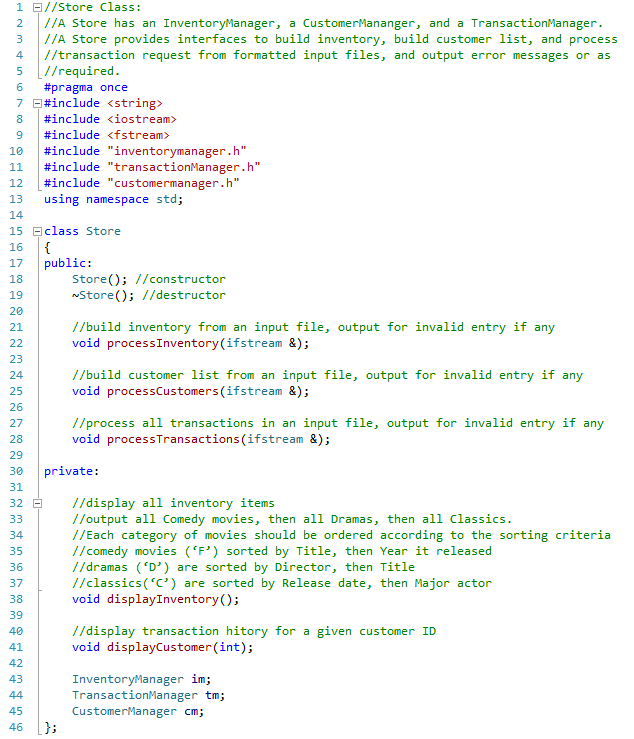
## **Class Diagram**

The architecture applied for the software is Objected-Oriented Model. We chose this architecture because it is the most appropriate model to describe the situation of the cd rental store.There are four main classes in our design: Store, InventoryManager, CustomerMananger, and a TransactionManager.

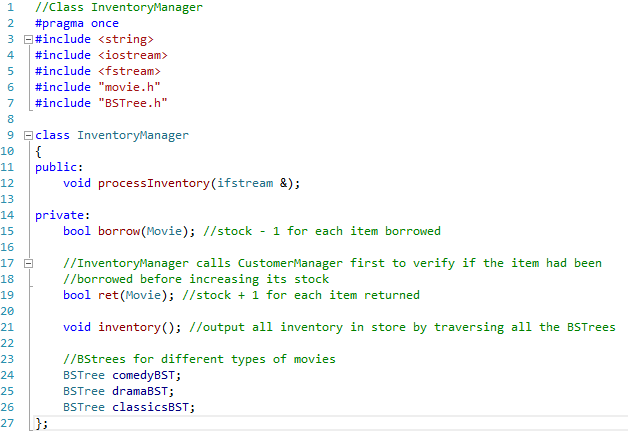


## **Class Description**

### **- Store**

Store class represents the store entity. It contains an InventoryManager, a CustomerMananger, and a TransactionManager. A Store provides interfaces to build inventory, build customer list, and process transaction request, and output error messages or as required.

### **- InventoryManager**

An InventoryManager contains a collection of store Items in ***binary search trees*** for quick look-up based on different sorting criteria.

#### **2.2.1 MovieFactory**

#### **2.2.2 Movie**

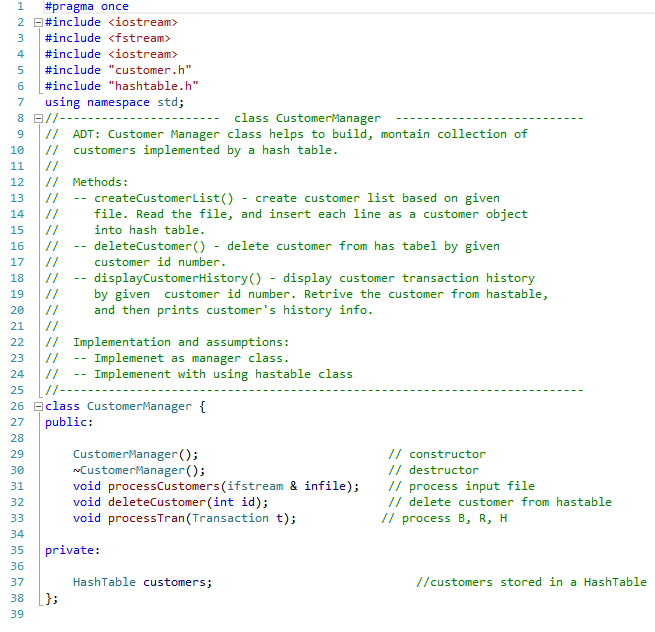
#### **2.2.3 Classic**

#### **2.2.4 Comedy**

#### **2.2.5 Drama**

#### **2.2.6- BSTree**

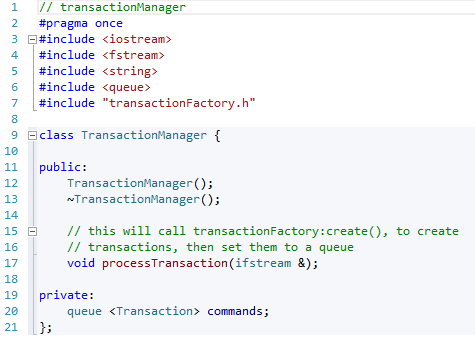
### **2.3 - CustomerManager**

* A CustomerManager maintains a collection of Customers implemented by a hash table, identified by their 4-digit Customer ID. CustomerManager adds and deletes Customers.
* CustomerManager is responsible for maintaining each Customer account.
* CustomerManager can also process a formatted file containing customer information.

#### **2.3.1 Customer**

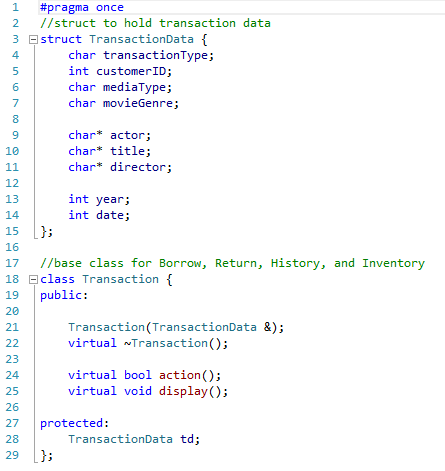
#### **HashTable**

### **2.4 - TransactionManager**

* A TransactionManager provides interface for processing Transaction requests.
* A TransactionManager can process a single Transaction or a batch of Transactions such as an input file, by calling the InventoryManager to update stock and calling the CustomerManager to update individual Customer account.
* A TransactionManager also maintains a queue for transaction output.

#### **2.4.1 TransactionFactory**

#### **2.4.2 Transaction**

Transaction is the base class for Borrow, Return, Inventory, and History.

#### **2.4.3 Borrow**

#### **2.4.4 Return**

#### **2.4.5 History**

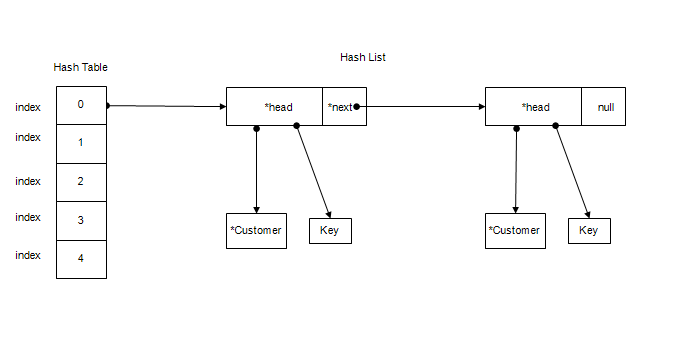
#### **2.4.6 Inventory**

## **Memory Diagram**

### **3.1 Hash Table**

Chain hashtable \* Customer Node structure diagram.

This data structure used in CustomerManager to keep track of all customers that the store currently has. For design, we plan use one HashTable for easy access for any customer. This implementation will reduce searching time of given customer.

Cha9j

### **3.2 BSTree**

