CSS 343

Assignment 4 Design Document

Team --: Cuong Vo, Jessela Budiman, Quan Nghiem

Overview: This is a short description of the design and how the pieces fit together (the interaction between the classes). Include a description of main (List the objects that you have. Main should be short.)

This Assignment4 design document includes all the required components and serves as a model for our implementation for movie rentals. The Store class contains the different lists of object such as Transaction, Customer(stored as ID), and Product (Movie) class and all functions to manage and modify these objects. A hash table which stores all the product that has a max size of 26 for A-Z letters, which maximize the finding efficiency. Products are an object which an array hold all the products (for extendability of DVD,VCR, music). Currently, we have movies that inherit children polymorphism types: comedy, drama, and classic, and has this list in a BinaryTree for sorting, deleting, and inserting. Each customer has their own unique ID, name and history transaction vector. Transaction inherits children classes Borrow, Return and History per the customer command. Our main reads the given text files to set the customers, products, and transactions in the created object Store teamMinusStore("Team--").

Class descriptions: For each class in the design, describe the data and methods as part of a documented C++ header file (**Exception**: you do not need to include h files of classes you will not implement, of the extensions beyond the assignment specifications, but you must include a description of those classes). The task that each function performs and the purpose of each data member should be clearly described. High-level pseudo code should be included for the most important methods (for example, those that control the flow of the program). Not all parameters need to be included for methods. Please order the files properly, i.e., put the most important classes first, put parent classes before children classes.

- **Header Files**
-
- #include <fstream>
- #include <vector>
- #include "customer.h"
- #include "hashtable.h"
- #include "bintree.h"
- //--
- // Class: Store
- // Objcet contains, customers, Product, transactions, and all functions that
- // help with create, modify, and manipulate all data involving these objects.
- //--

```
-	const int MAXCUSTOMERS = 10000;
-	const int  MAX_PRODUCT = 26;
-	using namespace std;
-
-	class Store {
-	public:
-	  // constructor
-		Store();
-		Store(string );
-		Store(const Store& );
-		~Store();	//destructor
-	  // Create a news Customer and Product
-		void createCustomers(istream&);
-		void createProduct(istream&);
-	  // Process all the Transaction in istream
-		void processTransactions(istream&);
-	  // display functions
-		void display();
-		void displayProduct() const;
-		void displayCustomerBase() const;
-		bool lookUpCustomer(int) const; // look up customer by ID
-		string getStoreName() const; // get the name of the Store
-		Customer getCustomer(const int) const;
-	private:
-		Customer customerList[MAX_CUSTOMERS]; // custer list
-		BinTree productList[MAX_PRODUCT]; // product/movie list
-		vector<Transaction> storeTransactionHistory; // list of all transaction
-		HashTable Table; // hash table to find product easier
-		string storeName; // name of the store.
-	};
-
-	#include "transaction.h"
-	//-------------------------------------------------------------------------
-	// Class: Borrow
-	// Child of Transaction class. Each object contains all data
-	//  members contained in a transaction object and functions needed
-	//  to modify data and display a borrow object.
-	//-------------------------------------------------------------------------
-	class Borrow : public Transaction {
-	public:
-	  //constructors
-		Borrow();
-		Borrow(const Borrow& );
-		~Borrow(); // destructor
-	  //mutator
-		virtual bool setData(string , Product * , Customer * );
```

```
-               virtual void display() const;
-               virtual Transaction * create();
-       };
-
-       /*-------------------------------------------------------------------------
-         file comedy.h
-         Specific movie class for movies belonging to the Comedy category.
-         Assumptions:
-           -- Comedy is represented by the letter ‘F’ in the data file
-               -- Sorted by title and then date.
-       -------------------------------------------------------------------------*/
-       #pragma once
-       #include "movie.h"
-       class Comedy : public Movie {
-       public:
-       // Constructors
-               Comedy();
-               ~Comedy();
-       // for command data file
-               virtual void setCmdData(istream&);
-       // instantiates a Comedy object
-               virtual Product* create() const;
-               virtual bool operator<(const Product&) const;
-               virtual bool operator==(const Product&) const;
-       };
-       /*-------------------------------------------------------------------------
-         file classic.h
-         Specific movie class for movies belonging to the Classic category.
-         Implementation:
-           -- Adds members for release month and famous actor
-           -- Classics are represented by a ‘C’
-                -- Sorted by release date and then famous actor.
-       -------------------------------------------------------------------------*/
-       #pragma once
-       #include "movie.h"
-       class Classic : public Movie {
-       public:
-       // Constructors
-               Classic();
-               ~Classic();
-       // the two different setData funcs for different text file reads
-               virtual void setData(istream&);
-               virtual void setCmdData(istream&);
-       // displays for body and header
-               virtual void display() const;
-               virtual void displayHeader() const;
```

```
-       // creates the Classic object
-               virtual Product* create() const;
-               virtual bool operator<(const Product&) const;
-               virtual bool operator==(const Product&) const;
-       private:
-       // unique attributes for Classic movies
-               int month;
-               string actorFirst, actorLast;
-       };
-       #include <iostream>
-       #include <string>
-       #include <vector>
-       #include "transaction.h"
-       using namespace std;
-       //-------------------------------------------------------------------------
-       // Class: Customer
-       // Object hold information used to identify a customer, rental information,
-       // transaction information, and history information. Objects are
-       // created from text file and stored in array inside of store.
-       //-------------------------------------------------------------------------
-       class Customer {
-       public:
-         //constructors
-               Customer();
-               Customer(istream&);
-               Customer(const Customer&);
-         //destructor
-               virtual ~Customer();
-         //mutator
-               void setData(istream& infile);
-         //accessor
-                int getCustomerID() const;
-                string getFirstName() const;
-                string getLastName() const;
-               void displayCustomerHistory() const; //display all Customer'sHistory
-               void addTransaction(Transaction ); // add a Transaction
-               virtual void display() const; // diplay everything
-         //operator
-               virtual bool operator==(const Customer& rhs) const;
-               virtual bool operator!=(const Customer& rhs) const;
-       private:
-               int id;
-               string firstName;
-               string lastName;
-               vector<Transaction> transactionHistory;
-       };
```

```
-       /*-------------------------------------------------------------------------
-         file drama.h
-         Specific movie class for movies belonging to the Drama category.
-         Assumptions:
-           -- Sorted by director and then title
-           -- Classics are represented by a ‘D’
-       -------------------------------------------------------------------------*/
-       #pragma once
-       #include "movie.h"
-       class Drama : public Movie {
-       public:
-       // Constructors
-                Drama();
-                ~Drama();
-       // for command data file
-                virtual void setCmdData(istream&);
-       // instantiates a Drama object
-                virtual Product* create() const;
-                virtual bool operator<(const Product&) const;
-                virtual bool operator==(const Product&) const;
-       };
-
-       #include "transaction.h"
-       //-------------------------------------------------------------------------
-       // Class : History
-       // child of transaction, displays the history for specific customer.
-       //
-       //-------------------------------------------------------------------------
-       class Customer;
-       class History : public Transaction{
-       public:
-         //constructor
-                History();
-         //copy constructor
-                History(const History& rightSide);
-         //destructor
-                ~History();
-         //mutator
-                virtual bool setData(string, Product *, Customer *);
-                virtual Transaction * create();
-
-       };
-       #include <iostream>
-       #include <string>
-       #include "classic.h"
-       #include "drama.h"
```

```
-       #include "comedy.h"
-       #include "transaction.h"
-       #include "borrow.h"
-       #include "return.h"
-       #include "history.h"
-       class Movie;
-       class Transaction;
-       //---------------------------------------------------------------------------
-       // Class: Factory
-       //  This class contain all the functions that is needed for building
-       // inventory and transaction objects. Uses predefined indexes for each  type
-       // and predefined maximum number of items per hash table.
-       //---------------------------------------------------------------------------
-       using namespace std;
-       class HashTable
-       {
-          public:
-             static const int MAX_SIZE = 26; //Number of letter in alphabet
-             HashTable();
-             ~HashTable();
-
-             Product* createMovie(char, istream&); //create new Product
-             Transaction* createTransaction(char,istream&); //create new Transaction
-
-             int getSubscript(char); //get array subscript from letter
-             string getMediaType(char); //get mediatype from letter
-
-          private:
-             //Depending on the subscript returned from letter, factory will
-             //create the specified object if it exists or return NULL
-                  Product* movieInventory[MAX_SIZE];
-                  Transaction* transactionInventory[MAX_SIZE];
-                 string mediaType[MAX_SIZE];
-
-             int hash(char); //return int subscript from char a-0, b-1 , and so on
-             void initProduct()); //set initial values in arrays
-       };
-       /*---------------------------------------------------------------------------
-         file movie.h
-         Pure virtual Movie class that keeps track of director, title, and year
-         Implementation:
-           -- Adds director, title, and year of release on top of name
-           -- Still pure virtual for some function, defined later by child classes
-                   -- Contains some default functions to be used by child classes
-       ---------------------------------------------------------------------------*/
-       #pragma once
```

```
-	#include "product.h"
-	class Movie : public Product {
-	public:
-	// Constructors
-		Movie();
-		virtual ~Movie();
-	// sets the data from a file of movies
-		virtual void setData(istream&);
-	// sets the data from a file of commands
-		virtual void setCmdData(istream&) = 0;
-	// the two displays that output the inventory
-		virtual void display() const;
-		virtual void displayHeader() const;
-	// makes the object
-		virtual Product* create() const = 0;
-	// check for smaller than and equal to between products, for the BST
-		virtual bool operator<(const Product&) const = 0;
-		virtual bool operator==(const Product&) const = 0;
-	// gets the product title
-		virtual string getProd() const;
-	// getters
-		string getDirector() const;
-		string getTitle() const;
-		int getYear() const;
-
-	protected:
-		static const int MAX_CHARS_TITLE = 21;    // max length to display
-		static const int MAX_CHARS_DIRECTOR = 15; // max length to display
-		string director, title, genre;           // the needed variables
-		int year;
-	};
-	/*-----------------------------------------------------------------------
-	  file product.h
-	  Pure virtual class representing a general product, controls distribution
-	  Assumptions:
-	    -- Controls the product distribution of the copies
-	-----------------------------------------------------------------------*/
-	#pragma once
-	#include <iomanip>
-	#include <iostream>
-	#include <string>
-	using namespace std;
-	const int NONE = 0;  // for checking zero, or no copies
-	class Product {
-	public:
-	// Constructors
```

```
-               Product();
-               virtual ~Product();
-       // two setData, one for product data, other is for command data
-               virtual void setData(istream&) = 0;
-       virtual void setCmdData(istream&) = 0;
-       // two displays, for body and header, resp.
-               virtual void display() const = 0;
-               virtual void displayHeader() const = 0;
-       // instantiates a product object
-       virtual Product* create() const = 0;
-               virtual bool operator<(const Product&) const = 0;
-               virtual bool operator==(const Product&) const = 0;
-       // getter for the product type
-               virtual string getProd() const = 0;
-       // -- Product class functions --
-               // sets the max amount for copies in
-                       void setMaxCopy(const int);
-               // increments copies by one
-                       void increaseCopies();
-               // decrements copies by one
-                       void decreaseCopies();
-               // getter for amounts in/out
-                       int getAmountIn() const;
-                       int getAmountOut() const;
-       private:
-               int maxCopies;   // limit of copies in at one time
-               int amtOfCopies; // copies actually in at one time
-       };
-       #include "transaction.h"
-       //--------------------------------------------------------------------------
-       // Class: Return
-       // return object, child of transaction. Each object contains all data
-       //  members contained in a transaction object and functions needed
-       //  to modify data and display a return object.
-       //
-       //--------------------------------------------------------------------------
-       class Return : public Transaction {
-       public:
-         //constructors
-               Return();
-               Return(const Return&);
-               virtual ~Return(); //destructor
-         // mutator
-               virtual bool setData(string, Product *, Customer *);
-               virtual void display() const;
-               virtual Transaction * create(); //create new return object
```

```
-       };
-       #include <movie.h>
-       //---------------------------------------------------------------------------
-       // Class: Factory
-       //This transaction class contains all the functions that modify and display
-       // all the transaction happens in the store. Transaction has a Product
-       // point to specific movie, so we don't need time to find it. It also has
-       // transactionType.
-       //---------------------------------------------------------------------------
-       class Transaction {
-       public:
-               Transaction(); // constructor
-               Transaction(const Transaction& ); //copy constructor
-               virtual ~Transaction(); // destructor
-         //mutator
-               virtual bool setData(string media, Product * item, Customer * aCustomer);
       // setData
-               virtual void display() const;// display the transaction
-               virtual Transaction * create();
-         //accessor
-               string getMediaType(); //get media type
-               string getTransactionType();         // get transaction type
-               Product * getItem() const;  //pointer to Product Item
-
-       protected:
-               string transactionType;
-               string mediaType;
-               Product * product;
-
-       };
```