 **COSC 1320**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Seat # \_\_**

**Estimated Hours 2**

**Actual Hours**

**TA (Textual Analysis) for UML MVC CLASS Diagram MODEL**

**Programming Assignment 4:**

**(10 points)**

**(Due date found in the COSC 1320 BB Calendar!)**

**You must use Microsoft WORD program. Insert the TEMPLATEs for Actors and Use Cases in this Requirements Document.**

*Please have Step 1, 2, 3, 4, and 5 in this order.*

PLEASE use the “TA for UML MVC Class Diagram MODEL Movie Company System.doc” AS TEMPLATE.

(Do not Show STEP 1,…; do not change Line Numbers; do not change Number of Pages)

Any DIAGRAM that is NOT the result of CUT and PASTE

WILL BE IGNORED. (YOU WILL GET ZERO POINTS)

**Requirements Analysis**: Construct the **UML** **MVC** **CLASS** Diagram MODEL

Controller

ProgrammingAssignment4 **software product** **for a Video Store**

For a family or an individual, a favorite place to go on weekends or holidays is to a video store to rent movies. A new video store in your neighborhood is about to open. However, it does not have a program to keep track of its videos and customers. The store managers want someone to write a program for their system so that the video store can function. The program should be able to perform the following operations:

Controller: 1: displayInventory(): void

1. **Display** a list of **all** of the videos in the store (full information).

Controller: 2: searchInventory(): void

2. **Search** a video **- by title** and show the details of **a** particular video (title, stars (up to two!), producer, director, productioncompany, copiesinstock).

Controller: 3: rentVideo(string): void

3. **Rent** a video.

Controller: 4: displayCustomers (): void

4. **Display** a list of **all** of the store’s customers.

Controller: 5: displayCustomerInfo(): void

5. **Search** a customer by name and display the details of **a** particular customer **– by last name** (firstname, lastname, id).

6. **Display** a list of all of the videos rented by **a** particular customer (firstname, lastname, title, title).

Controller: 6: displayCustomerRecord(): void

ProgrammingAssignment4 should illustrate the **O**bject **O**riented **P**aradigm methodology and, in particular, **inheritance** and **overloading**.

There is a need to maintain the following lists (**cannot use STL Classes**):

 A list of all of the videos in the store

 A list of all of the store’s customers

 Lists of the videos currently rented by the customers

Controller: 7: readFile(): void

Provide **member functions** to load the video store collection, customer information & customer rental information from the file “**Programming Assignment 4 Data.txt**” (FILE INPUT) into your **ProgrammingAssignment4** **APPLICATION** memory (**please DO NOT read from File and write to File skipping reading into the program memory**) and write it back out to the **SAME** data file (FILE OUTPUT) when exiting from **ProgrammingAssignment4** **APPLICATION** **“Programming Assignment 4 Data.txt”**.

Controller: 8: writeFile(): void

View

View: 1: showMenu(): void

VideoStoreDriver: 1: cont: Controller

VideoStoreDriver

VideoStoreDriver: 2: view: View

VideoStoreDriver: 1: main(): int

The data in the input file **“Programming Assignment 4 Data.txt”** is in the following format:

Video: 1: Video()

Video: 2: Video(string,string,string,string,string,string,int)

Video: 3: getTitle(): string

Video: 4: getStar1(): string

Video: 5: getStar2(): string

Video: 6: getProducer(): string

Video: 7: getDirector(): string

Video: 8: getCompany(): string

Video: 9: getQuantity(): string

Video: 10: setTitle(): void

Video: 11: setStar1(): void

Video: 12: setStar2(): void

Video: 13: setProducer(): void

Video: 14: setDirector(): void

Video: 15: setCompany(): void

Video: 16: setQuantity(): void

Video: 17: toString(): string

Video

Controller: 1: vidCtr: int

**number of videos**

Video: 1: title: string

video title ( that is, the name of the movie)

Video: 2: star1: string

movie star1

Video: 3: star2: string

movie star2

Video: 4: producer: string

movie producer

Video: 5: director: string

movie director

Video: 6: prodCo: string

movie production co.

Video: 7: copies: int

number of copies

. . .

Customer:1: fName: string

Customer

**Customer**’s first name**,**last name**,**and id

Customer:4: amountRented: int

Customer:3: id: int

Customer:2: lName: string

. . .

**Customer** id**,**video title**,**video title

Customer:5: currentRentals: string[]

A sample input file “**Programming Assignment 4 Data.txt**” follows:

Customer: 1: Customer()

Customer: 2: Customer(string,string,int)

Customer: 3: getFirstName(): string

Customer: 4: getLastName(): string

Customer: 5: getID(): int

Customer: 6: setFirstName(): void

Customer: 7: setLastName(): void

Customer: 8: setID(): void

Customer: 9: addRental(): void

Customer: 10: deleteRental(): void

Customer: 11: printRentals(): void

Customer: 12: toString(): void

**3**

Titanic

Kate Winslet

Leonardo DiCaprio

Cameron

Cameron

20th Century Fox

**2**

Jerry MacGuire

Tom Cruise

Renne Zellwegger

Brooks

Crowe

United Artists

**18**

Rain Man

Dustin Hoffman

Tom Cruise

Johnson

Levinson

United Artists

Controller: 3: custArr[]: Customer\*

**2**

**Donald,Duck,1**

**Mickey,Mouse,2**

**Minnie,Mouse,3**

**Goofy,Dog,4**

Controller: 4: vidArr[]: Video\*

**4,Rain Man,Titanic**

**1,Jerry MacGuire**

Create a **ProgrammingAssignment4.cpp** that contains the **main ()**.

**Class**es must start with UpperCase Letter.

**Class**es must have **.h** and **.cpp**.

One **Class** per File Name that matches the **Class** Name.

**You cannot use any DATA STRUCTURES STL classes! Use arrays.**

**One Class per .h &.cpp file** (you will lose 15 points if not)**!**

**Must use Inheritance** (you will lose 20 points if not)**!**

**Each Class MUST HAVE a toString() member function**(you will lose 20 points if not)**!**

**Is toString() member function used** to display customers, videos, or rented videos by customer (you will lose 10 points if not)**!**

Each .h & .cpp file must have these 4 lines of comments (you will lose 5 points if not):

// COSC 1320 Summer 2015

// Name:

// Programming Assignment 4

// This is my own work; I will not post

|  |
| --- |
| View |
|  |
| + showMenu( ) : void |

has a

|  |
| --- |
| VideoStoreDriver |
| - cont: Controller  - view: View |
| + main ( ) |

|  |
| --- |
| Controller |
| - vidCtr: int //1  - custCtr: int //2  - custArr[]: Customer\* //3  - vidArr[]: Video\* //4 |
| + displayInventory(): void //1  + searchInventory(): void //2  - rentVideo(): void //3  + displayCustomers (): void //4  + displayCustomerInfo(): void //5  + displayCustomerRecord(): void //6  + readFile(): void //7  + writeFile(): void //8 |

|  |
| --- |
| Video |
| - title: string //1  - star1: string //2  - star2: string //3  - producer: string //4  - director: string //5  - prodCo: string //6  - copies: int //7 |
| + Video() //1  + Video(string,string,string,string,string,string,int) //2  + getTitle(): string //3  + getStar1(): string //4  + getStar2(): string //5  + getProducer(): string //6  + getDirector(): string //7  + getCompany(): string //8  + getQuantity(): string //9  + setTitle(): void //10  + setStar1(): void //11  + setStar2(): void //12  + setProducer(): void //13  + setDirector(): void //14  + setCompany(): void //15  + setQuantity(): void //16  + toString(): string //17 |

has a

uses

uses

|  |
| --- |
| Customer |
| - fName: string //1  - lName: string //2  - id: int //3  - amountRented: int //4  - currentRentals: string[] //5 |
| + Customer() //1  + Customer(string,string,int) //2  + getFirstName(): string //3  + getLastName(): string //4  + getID(): int //5  + setFirstName(): void //6  + setLastName(): void //7  + setID(): void //8  + addRental(): void //9  + deleteRental(): void //10  + printRentals(): void //11  + toString(): void //12 |

|  |
| --- |
| VideoStoreDriver |
| - cont: Controller  - view: View |
| + main ( ) |

int main(){

creates instances of Controller and View for the actor to engage.

Will call showMenu() in View to prompt for user input.

After getting input, the controller, performs the necessary actions.

Repeats until the user chooses to save and exit the program.

|  |
| --- |
| View |
|  |
| + showMenu( ) : void |

}

void showMenu(){

displays the menu.

}

|  |
| --- |
| Controller |
| - vidCtr: int //1  - custCtr: int //2  - custArr[]: Customer\* //3  - vidArr[]: Video\* //4 |
| + displayInventory(): void //1  + searchInventory(): void //2  + rentVideo(string): void //3  + displayCustomers (): void //4  + displayCustomerInfo(): void //5  + displayCustomerRecord(): void //6  + readFile(): void //7  + writeFile(): void //8 |

void displayInventory(){

displays the contents of vidArr

}

void searchInventory(){

asks cashier for a title and loops through vidArr to see if it’s available.

If there is still stock of that video, rentVideo(string) is called

}

void rentVideo(string){

asks cashier for a customer id and adds video to their record. Decrements copies for that video

}

void displayCustomers (){

loops through custArr and displays customers

}

void displayCustomerInfo(){

asks cashier for last name and displays that customer’s info

}

void displayCustomerRecord(){

asks cashier for last name and displays that customer’s rentals

}

void readFile(){

reads file to build inventory, customer list and current rentals

}

void writeFile(){

outputs all current data to the same file

}

|  |
| --- |
| Video |
| - title: string //1  - star1: string //2  - star2: string //3  - producer: string //4  - director: string //5  - prodCo: string //6  - copies: int //7 |
| + Video() //1  + Video(string,string,string,string,string,string,int) //2  + getTitle(): string //3  + getStar1(): string //4  + getStar2(): string //5  + getProducer(): string //6  + getDirector(): string //7  + getCompany(): string //8  + getQuantity(): string //9  + setTitle(): void //10  + setStar1(): void //11  + setStar2(): void //12  + setProducer(): void //13  + setDirector(): void //14  + setCompany(): void //15  + setQuantity(): void //16  + toString(): string //17 |

Video(){

Initializes all values to default

}

Video(string,string,string,string,string,string,int){

Initializes values with provided arguments

}

Setters and getters(){

Set and return the value described by the name of the function

}

void toString(){

returns a string of the user’s data

}

|  |
| --- |
| Customer |
| - fName: string //1  - lName: string //2  - id: int //3  - amountRented: int //4  - currentRentals: string[] //5 |
| + Customer() //1  + Customer(string,string,int) //2  + getFirstName(): string //3  + getLastName(): string //4  + getID(): int //5  + setFirstName(): void //6  + setLastName(): void //7  + setID(): void //8  + addRental(): void //9  + deleteRental(): void //10  + printRentals(): void //11  + toString(): void //12 |

Customer (){  
Initializes all values to default  
}

Customer (string,string,string,string,string,string,int){  
Initializes values with provided arguments  
}

Setters and getters(){  
Set and return the value described by the name of the function  
}

void addRental(){  
adds movie and increments amountRented;  
}

Void deleteRental(){  
Deletes movie, shifts array and decrements amountRented;  
}

void toString(){  
returns a string of the user’s data  
}