Assignment 4 - Design Document

Prarin Behdarvandian
Regina A Bloomstine
Helen Mirabella
Brent Stone

ReadME.txt:

The purpose of this project is to create a program for a **movie rental store**, in order for them to store and keep track of their customers transactions and be up to date with movie inventory. Some of the specifications are as follows:

The store would consist of multiple classes, related to each over and reused with **polymorphism** and **inheritance**, as showed on the **class diagram** on the following page.

The data structure used for the customer database is a **Hash Table**. Our proposal is for the hash table to consist of an array of linked lists. The assignment was chosen because of its efficiency for searching and retrieving.

The movie data base the structure used is a **B Tree with three pointers:** left, right and middle. We felt that it was the best structure due to the number of genera the video store is starting with. If, at any point, other genera would be added, the B tree will be able to adapt to the changes. However, our second option, which we are planning to adapt later, would be using **3** - or more - **BSTrees**: one for each movie category.

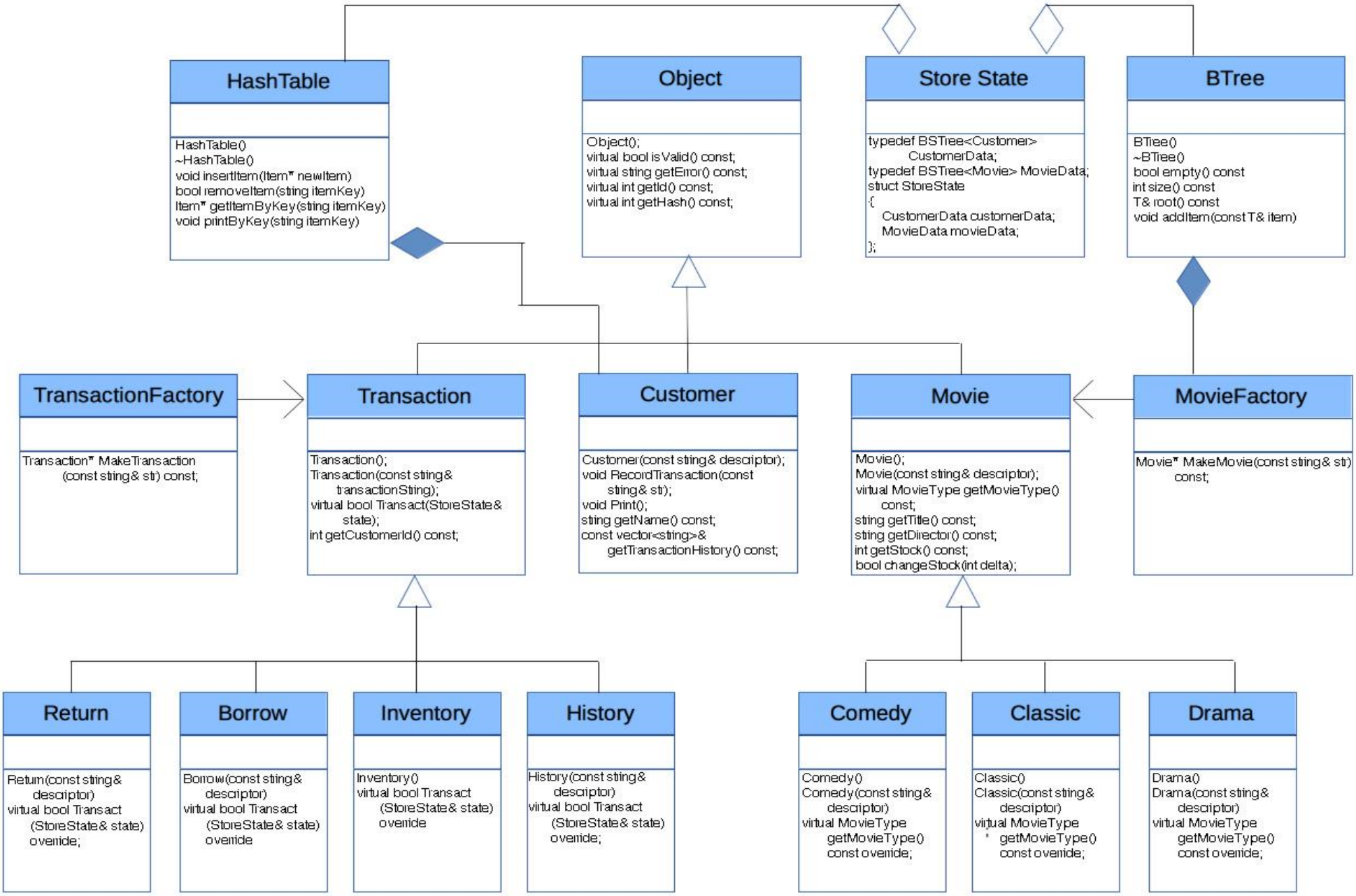
Transaction and Movie classes are using the **Factory design pattern**. Due to the architecture of our program the factory pattern allows us to encapsulate the object creation process. This also allows us to follow the open/closed principle – open for extension closed for modification, in both the movie and transaction classes.

Movie and **Transactions** types have their own separate classes in order to be reused.

History of transactions could be accessed at any time.

Search and **sorting** is currently done by the year of the making.

Class Diagram:



Class Descriptions:

A - Rental Store Class

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Rental Store Class:
5   //This Rental Store Class reads consist of three functions that processes 3 seperate infiles.
6   //The three file that will be read are data4movies.txt, data4customers.txt and and data4commands.txt
7   //The functions will build our movie inventory, customer database and process all listed transactions
8    #pragma once
9   #include <iostream>
10  #include "BTree.h"
11  #include "HashTable.h"
12  #include "CustomerAcct.h"
13   using namespace std;
14
15  class RentalStore
16  {
17      RentalStore();  //Constructor
18      ~RentalStore(); //Destructor
19
20      void processMovieFile(ifstream &);      //builds movie inventory
21      void processCustomerFile(ifstream &);   //builds customer database
22      void processCommandFile(ifstream &);    //processes all transactions
23
24  private:
25      struct storeData {
26          BTree movieData;
27          HashTable customerTable;
28      };
29  };
```

Pseudo Code for Rental Store

```
void processCustomerFile(ifStream)
{
    while(not end of file)
    {
        string = getLine();
        parse string;
        create new customer object(number, firstname,lastname)
        hastable.insert(customer);

    }
}

void processMovieFile(ifStream)
{
    while(not end of file)
    {
        string = getLine();
        pass string to movie factory object;
            //factory will parse string to figure out what
            //child class to call
            //child class will generate specific object
        once object has been created, add to the data base
    }

}

void processTransaction(ifStream)
{
    while(not end of file)
    {
        string = getLine();
        pass string to movie transaction factory object;
            //factory will parse string to figure out what
            //child class to call
            //child class will generate specific object
        once object has been created, transaction will be executed
    }
}
```

B.1 – BTree:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //BTree Class:
5   //The Btree Class is the implementation of out data structure for the movie database.
6   //Each tree will first be sorted by the type of movie then by the title.
7   #pragma once
8   #include "Movie.h"
9   class BTree
10  {
11  public:
12      BTree();    //Constructor
13      ~BTree();   //Destructor
14
15      bool insert(const Movie *insert);       //inserts movie object into Btree
16      bool retrieve(const string &name, const Movie *& mMovie); //retrieves movie object
17      void empty();       //deletes tree
18
19  private:
20      //Btree Node
21      struct Node {
22          Node* left = NULL;
23          Node* middle = NULL;
24          Node* right = NULL;
25
26          Movie* data;
27      };
28      Node* root = NULL;
29  };
```

Pseudo Code for BTree
Note - We have decided to change our data structure to a BSTree

Bool Insert:

- Each genre type has their own BSTree
- It will be sorted by Title
    - If title and all other variables are similar except copies variable
        - Add 1 to the copies variable.
    - If same title but other variables are different
        - Add it as its child

Bool retrieve:

- Find the class genre
- Look by year and then by title
- And then output it to the user

Void Empty:

- Empty Tree

B.2 – Movie Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Movie Class:
5   //This is the base class for the 3 different types of movies - Comedy, Classic, Drama
6   #pragma once
7   #include <string>
8   using namespace std;
9
10  class Movie
11  {
12  public:
13      Movie();            //Constructor
14      virtual ~Movie();   //Destructor
15
16  private:
17      //Parent Movie Data
18      string mDirectoFirst;
19      string mDirctorLast;
20      string mTitle;
21      int mYear;
22      int mCopies;
23      string mMediaType = "D";
24  };
25
26
```

B.3 - Movie Factory Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Movie Factory Class:
5   //The Movie Factory Class implements the factory pattern and will create a movie object
6   //This object will also be placed into the movie database after creation
7    #pragma once
8   #include <string>
9   #include "Movie.h"
10  #include "BTree.h"
11   using namespace std;
12
13  class MovieFactory
14  {
15  public:
16      MovieFactory(const string&);        //constructor with string
17      ~MovieFactory();                    //Destructor
18
19      Movie* createMovie(const string&); //Creates Movie Obejct using
20                                         //passed in string,
21  };
22
23
```

B.4 - Comedy Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Comedy Class:
5   //The Comedy Class inherits data from its parent class and sets the type to F;
6   #pragma once
7   #include "Movie.h"
8
9   using namespace std;
10
11  class Comedy : public Movie
12  {
13  public:
14      Comedy();           //Constructor
15      virtual ~Comedy();  //Destructor
16
17  private:
18      char mType = 'F';
19  };
20
21
```

B.5 – Classic Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Classic Class:
5   //The Classic Class inherits data from its parent class and sets the type to C.  It also
6   //has unique data that is initialized in the classic class
7   #pragma once
8   #include "Movie.h"
9   #include <string>
10
11  using namespace std;
12
13  class Classic : public Movie
14  {
15  public:
16      Classic();          //Constructor
17      virtual ~Classic(); //Destructor
18
19  private:
20      //Unique Data
21      char type = 'C';
22      int mMonth;
23      string mActorFirstName;
24      string mAcotrLastName;
25
26  };
27
28
```

B.5 – Drama Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Drama Class:
5   //The Drama Class inherits data from its parent class and sets the type to D;
6   #pragma once
7   #include "Movie.h"
8
9   using namespace std;
10
11  class Drama : public Movie
12  {
13  public:
14      Drama();            //Constructor
15      virtual ~Drama();   //Destructor
16
17  private:
18      char mType = 'D';
19  };
20
```

C.1 – Transaction Class

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Transaction Class:
5   //The Transaction Class creates a transaction from the string that is passed. Depending
6   //on the value, it will call different actions.  Borrow, Return, Inventory, or History
7   #include <string>
8   #include "RentalStore.h"
9
10  using namespace std;
11  class Transaction
12  {
13  public:
14      Transaction();                  //Constructor
15      Transaction(const string&);     //Constructor with string
16      virtual ~Transaction();         //Destructor
17      virtual bool TransAct(RentalStore &data);   //Action based on Trasaction Created
18      int getCustomerId() const;      //get customer ID
19  };
20
21
```

C.2 - Transaction Factory Class

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Transaction Factory Class:
5   //The Transaction Factory Class implements the factory pattern and will create a transaction
6   //object.  This object will also be processed after creation
7   #pragma once
8   #include "Transaction.h"
9   class TransactionFactory
10  {
11  public:
12      TransactionFactory();           //Constructor
13      virtual ~TransactionFactory();  //Destructor
14
15      Transaction* makeTransaction(const string&); //Creates Transaction ojbect
16  };                                               //with string that is passed
17
18
```

C.3 - Borrow Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Borrow Class:
5   //The Borrow Factory Class creates a Retrun object and manipulates the central database
6   //TransAct subtracts one from the inventory
7   #pragma once
8   #include "Transaction.h"
9   #include "RentalStore.h"
10
11  class Borrow : public Transaction
12  {
13  public:
14      Borrow(const string&);  //Constructor
15      virtual ~Borrow();      //Destructor
16
17      virtual bool TransAct(RentalStore &data);   //Action on database
18  };
19
20
```

C.4 - Return Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Return Class:
5   //The Return Factory Class creates a Retrun object and manipulates the central database
6   //TransAct adds one back to the inventory
7    #pragma once
8   #include "Transaction.h"
9   #include "RentalStore.h"
10
11
12  class Return : public Transaction
13  {
14  public:
15      Return(const string&);      //Constructor
16      virtual ~Return();                  //Destructor
17
18      virtual bool TransAct(RentalStore &data);   //Action on database
19  };
20
```

C.5 – Inventory Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Inventory Class:
5   //The Inventory Factory Class creates a Inventory object and prints the total inventory
6    #pragma once
7   #include "Transaction.h"
8   #include "RentalStore.h"
9
10  class Inventory : public Transaction
11  {
12  public:
13      Inventory();            //Constructor
14      virtual ~Inventory();   //Destructor
15
16      virtual bool TransAct(RentalStore &data);   //Action on database
17  };
18
19
```

C.6 – History Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //History Class:
5   //The Histor Class creates a History object and prints the actions that have occurred
6   //on a certain customer
7    #pragma once
8   #include "Transaction.h"
9   #include "RentalStore.h"
10
11
12  class History : public Transaction
13  {
14  public:
15      History(const string&);     //Constructor
16      virtual ~History();                 //Destructor
17
18      virtual bool TransAct(RentalStore &data);   //Action on database
19  };
20
```

D.1 – Hash Table Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //Hash Table Class:
5   //The HashTable Class creates a hash table, an array of linked list.  The functions that
6   //can be performed on the table is insert and retrieve.
7   #pragma once
8   #include "CustomerAcct.h"
9   class HashTable
10  {
11  public:
12      HashTable();    //Constructor
13      ~HashTable();   //Destructor
14
15      bool insert(CustomerAcct*&);                    //insert
16      bool retrieve(int acctNumber, CustomerAcct *&); //retrieve to edit
17
18  private:
19      //Node to place in array
20      struct Node {
21          Node* next;
22          CustomerAcct* data;
23      };
24      //Array of linked list
25      Node Htable[5];
26  };
```

D.2 – Customer Account Class:

```
1   //CSS 343 - Design Group 4
2   //Members - Prarin Behdarvandian, Regina A Bloomstine, Helen Roze Mirabella, Brent Stone
3   //
4   //CustomerAcct Class:
5   //The Customer Acct Class creates a customer account object to be stored in the hash table
6   //The unique feature of this class the storing the history of action of the customer
7   //as a linked list.
8   #pragma once
9   #include <string>
10  using namespace std;
11
12  class CustomerAcct
13  {
14  public:
15      CustomerAcct(int id, string firstName, string lastName);    //Constuctor
16      ~CustomerAcct();                                            //Destructor
17      int getID();                //returns customer ID
18      string getFirstName();      //returns customer first name
19      string getLastName();       //returns customer last name
20
21      bool recordTransaction(string transaction); //records the transaction
22
23  private:
24      struct Node
25      {
26          string mTrasaction;
27          Node *next = NULL;
28      };
29      Node *head = NULL;
30      int mID;
31      string mFirstName;
32      string mLastName;
33  };
```