**Insert Item**

Library.InitializeCatalog(LibraryItemInit.txt) {

LibraryItemFactory(Library); //instantiate LibraryItemFactory

LibraryItemFileReader(LibraryItemInit.txt); //Instantiate FileReader

Vector<vector<string> v = LibraryItemFileReader.Parse();

LibraryItemFactory.CreateAll(v);

}

LibraryItemFileReader.Parse() {

Parse through file;

Return vector<vector<string>>;

}

LibraryItemFactory.CreateAll(vector<vector<string>>) {

Foreach(vector<string> in vector<vector<string>>)

Create(vector<string>);

}

LibraryItemFactory.Create(vector<string>) {

itemType = vector[0];

Switch(ItemType) {

Case: Y

LibItem = new ChildrensBook(vector[0], vector[1], vector2, etc.);

Library.AddItem(LibItem);

Break;

Case…

Default…

}

}

Library.AddItem(LibraryItem) {

Switch (LibraryItem.itemType) {

Case(“Y”):

Add LibraryItem to Children’s Book Catalog

Break;

Case…

Default…

}

}

Catalog.Insert(LibraryItem) {

Index = hash(LibraryItem);

Array[index] = libraryItem;

}

**Check In Pseudo Code**

Library.ReadCommands(commands.txt) {

Librarian(Library); //instantiate Library

Librarian.getTransactionQueue(commands.txt);

Librarian.executeAll()

}

Librarian.getTransactionQueue(commands.txt) {

TransactionFactory(Librarian);

TransactionFactory.buildTransactionQueue(commands.txt);

}

CommandsFileReader.Parse() {

Parse through the file

return vector<vector<string>>

}

TransactionFactory.buildTransactionQueue(commands.txt) {

TransactionFileReader(commands.txt);

Vector<vector<string>> v = TransactionFileReader.Parse();

For (int I = 0; I < v.size(); v++) {

commandType = v[i][0];

Switch(commandType) {

Case(“C”):

tempLI = createTempLibItem(vector<string>);

tempPat = createTempPatron(vector<string>);

Library \* lib = Librarian.getLibrary();

CI = new CheckIn(tempLI, tempPat, lib);

Librarian.add(CI);

Break;

Case…..

Default…

}

}

}

TransactionFactory.createTempLibItem(vector<string> v) {

itemType = v[1];

switch (itemType) {

case “Y”:

return new ChildrensBook(v[1], v[2], v[3], etc.);

break;

case…

default…

}

}

TransactionFactory.createTempPatron(vector<string> v) {

Return new Patron();

}

CheckIn(LibaryItem\*, Patron\*, Library) {

Store arguments in private member variables.

}

Librarian.executeAll() { //execute all Transactions in the Queue<Transaction>

While(Queue<Transaction> is not empty) {

Transaction.execute();

}

}

CheckIn.execute() {

LI\* = Library.getLibraryItem(tempLibraryItem);

P\* = Library.getPatron(tempPatron);

If (LI\* != nullptr && P\* != nullptr) {

Library.checkIn(LI, P);

}

}

Library.checkIn(LibraryItem\*, Patron\*) {

LibraryItem->checkIn(Patron);

Patron->checkIn(LibraryItem);

}

LibraryItem.checkIn(Patron\*) {

Remove Patron from collection<Patron>;

Inventory++;

}

Patron.checkIn(LibraryItem\*) {

CheckOutRecord\* = getRecord(LibraryItem\*);

CheckOutRecord->checkedOut = false;

}