**Make BookBinTree an Abstract class and use pointers!**
**Create a Book pointer and instantiate it as a new Periodical, Fiction, or Childrens book**

- **Reuse the tree in 3 ways to hold the three different type of Books**

BookBinTree

Root : Book
struct Node: Book bookInfo, Node* left, Node* right

+BookBinTree()
+BookBinTree(const BookBinTree&)
+~BookBinTree()
+insert(Book*) : bool
+findBook(const Book&, Book *&) const : bool
+makeEmpty(): void
+isEmpty() const: bool
operator<<(ostream&) - (inorder trav.)

Book

-NumInLib : Int
-MaxNumInLib : Int

Title : String
Year : Int

+Book()
+~Book()

+checkOut(): void
+checkIn(): void
+operator=(const Book&) : Book&
+operator==(const Book&) const : bool
+operator!=(const Book&) const : bool
+operator<(const Book&) const : bool
+operator>(const Book&) const : bool
+operator<<(ostream&, const Book&) : ostream&
+getNumInLib() const: int
+setNumInLib(int): void

+create(): Book*
+setData(string) : void

-checkContent() const : bool

- **Create simply creates a new object w/ no data in it**
- **setData is what populates the object with the correct data**

create() returns a new Book object. Example:
**virtual Stuff* create() const { return new AObject; }**

checkContent - returns true if content is valid, returns false if not

Operators: Compared by date(year, then month) , then by title

Periodical

month : Int

Operators: Compared by author, then by title

Fiction

AuthorFirst : String
AuthorLast : String

Operators: Comapred by title, then by author

Childrens

AuthorFirst : String
AuthorLast : String

BookFactory

Book* book

+BookFactory()

+createBook(char) : Book*

- **createBook in factory calls on create() in Book class and takes in parameter that is the type of book!**

LibraryManager

BookFactory bf;

HashMap patrons

+LibraryManager()
+readInventory(ifstream&) : void
+readCustomer(ifstream&)
+readTransaction(ifstream&)

**Having a Transaction Factory does not comply with Command design pattern**

HashMap

patrons : Patron[]

-readFile(ifstream&)
+addPatron(Int, String, String)
+getPatron(Int)

-hashify(Int)

Checkout

book : Book*
customer* : Patron

Return

book : Book*
customer* : Patron

History

customer* : Patron

**Need some kind of data structure to store the transcations done for each customer**
- **idea: we could just use a vector??**

Transaction

+doTransaction(int, Book*) : virtual
+ display(): virtual
-readFile(ifstream&)

- **We are validating if the type of book is valid or not in the LibraryManager class**
- **All other validation is done in the book class itself**

Patron

customerID : Int
firstName : String
lastName : String

custHistory : vector<Transaction*>

+addToHistory(Transaction*)
+display()

+getHistory()

**added vector to store transaction history for each customer**

DisplayCatalogue

catalogue : array[BinTree]

HashTable for Customer Transcations
- column - patron #
- rows - transaction #