Jessica Nguyen

CSS343, Prog 4 Design

**Overview**

All objects in this assignment are: Driver class (includes main), Shop, Inventory List, Coin Inventory, Comic Inventory, SportsCard Inventory, Customer List, Customer, Item, Coin, Comic, and Sports Card.

The Shop class has an Inventory List and a Customer List.

Inventory List is like a shop’s item tracking system. It is a parent class that provides a template to organize Items. Inventory List has 3 children who represent Binary Search Trees for different Items: Coin Inventory, Comic Inventory, and SportsCard Inventory. Each BST stores Items. Item is a parent class that Coin, Comic and SportsCard inherit from.

Customer List is like a shop’s list of customers they have on file. Every customer has an ID associated with them for easier bookkeeping from the shop, even though customers themselves don’t know about their IDs. Customer List is a hash table of Customer objects with the key value being their ID numbers.

Basic structure in Main:

1. Create a Shop object
2. Pass inventory file to the Shop

/\* Read Inventory:

1. Instantiate an InventoryList object to keep track of all the Items in the file

2. Read each line in the inventory file and initialize all listed objects

\*/

1. Pass Customer file to the Shop

/\* Read Customer:

1. Create a CustomerList object to keep track of all the Customers in the file

2. Read each line in the customer file and initialize all listed customers

\*/

1. Pass Command file to the Shop

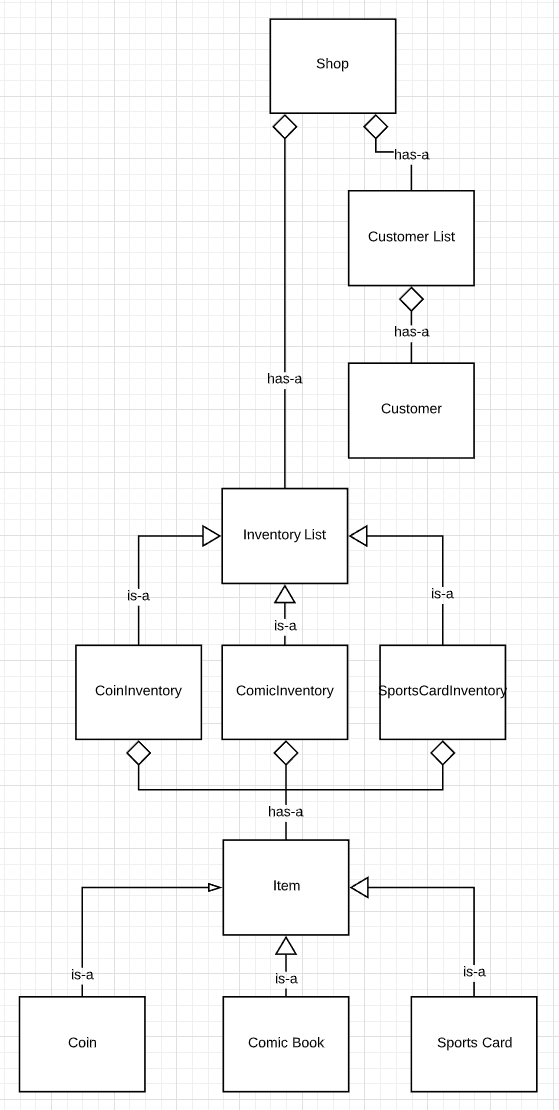
/\* Read Commands:

1. Read in each line to perform the action

2. Call Driver's readCommand() method to carry out appropriate actions

\*/

**UML Domain Model**

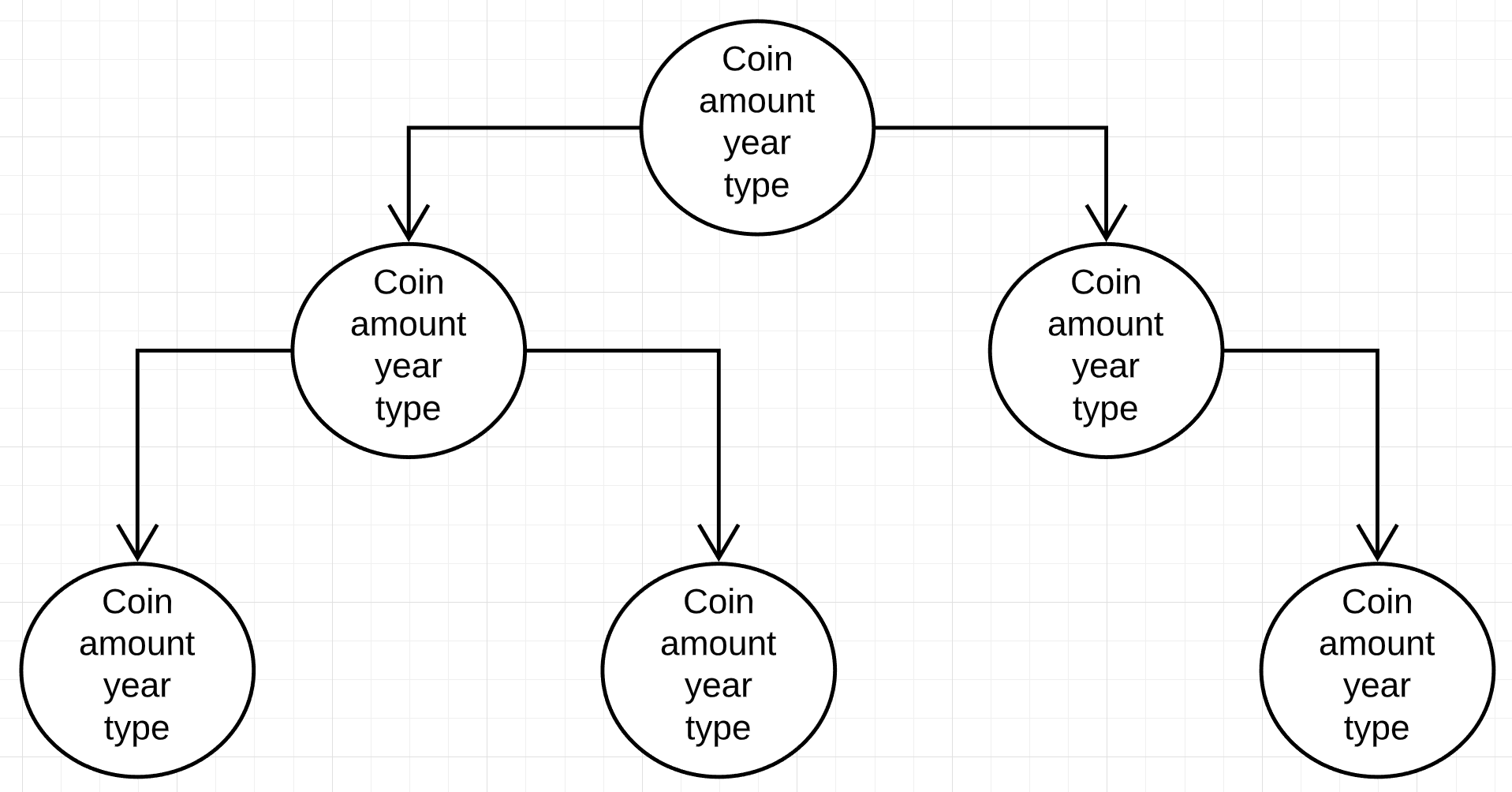
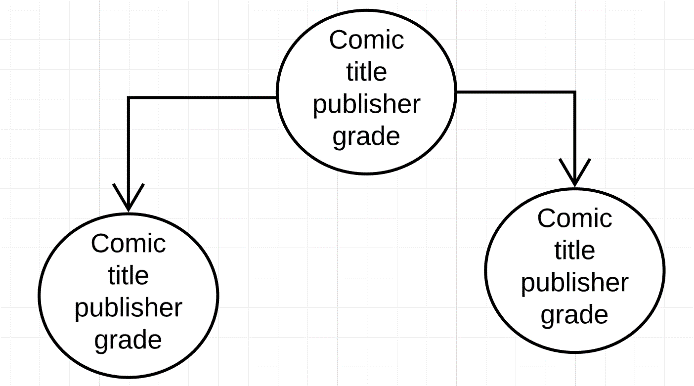


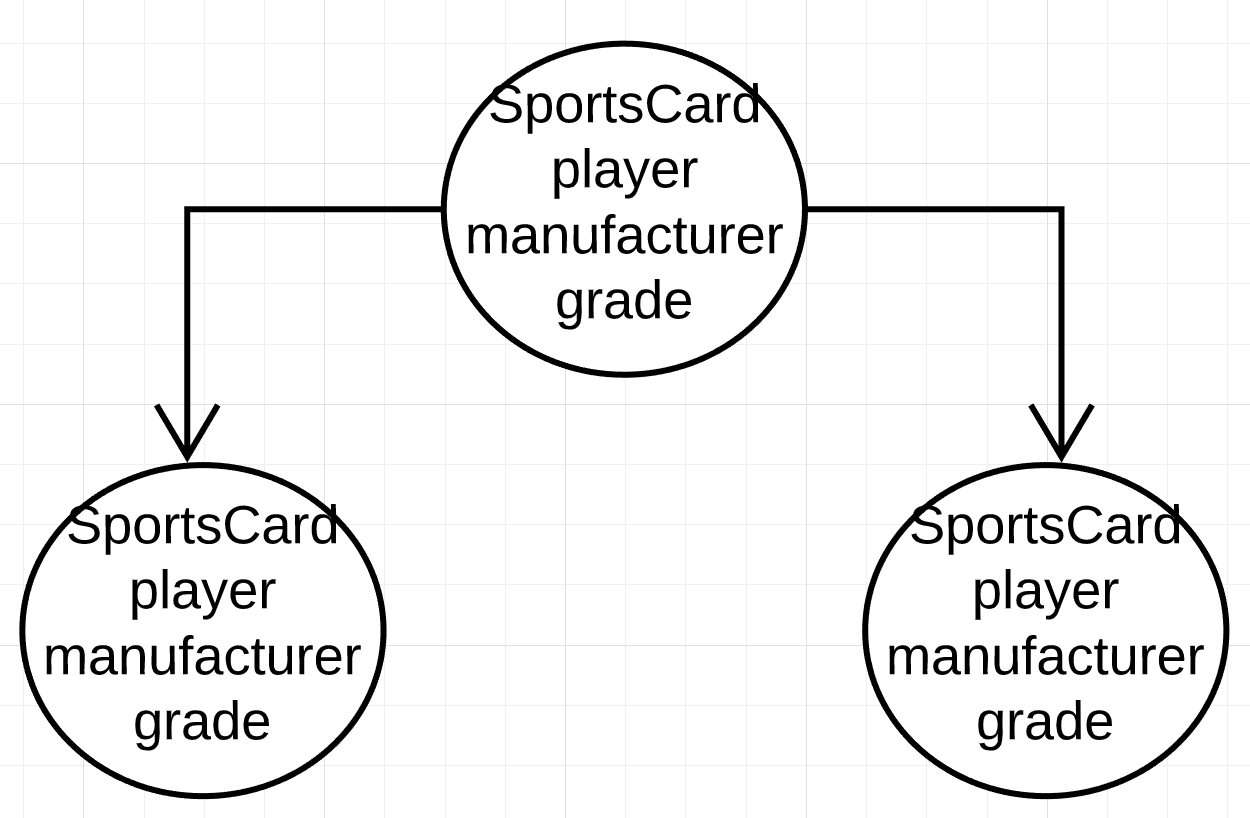
**Memory Diagram**

1. **In the InventoryList class**

Items are sorted in 3 separate Binary Search Trees depending on their item type; for example: the coinInventory sorts Coins by the order of type, then by year, then by grade.

This way, items can be sorted by the specified order, so that the driver can perform operations on the Items (change amount, add, remove, retrieve).



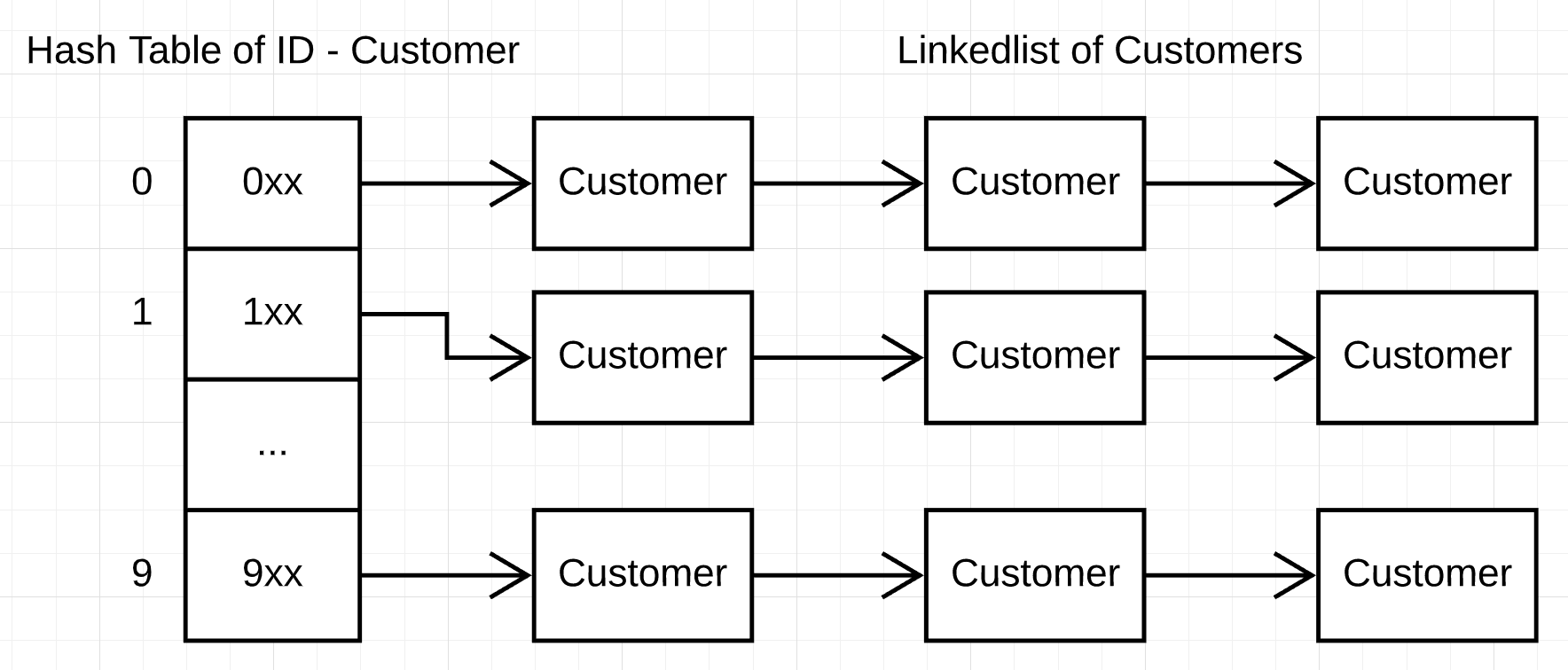


1. **In the CustomerList class**

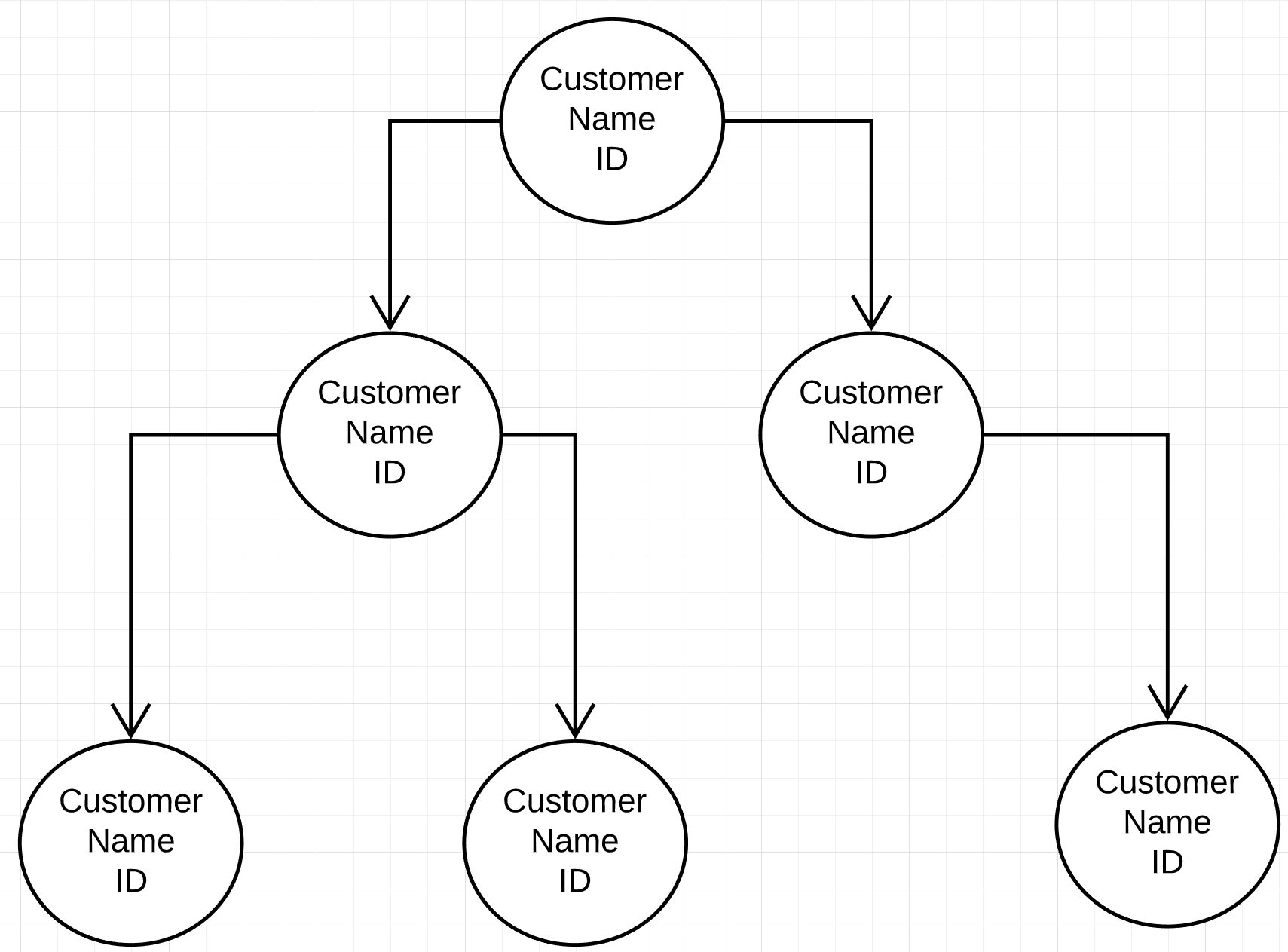
Customers are associated with their ID in a hash table. The hash function maps ID number to Customer object, the hashing method uses Open Hashing with linked list of Customers to store collided buckets.

Meanwhile, to sort Customer names in alphabetical order, I use a binary search tree to sort CustomerName nodes (each node has a string and int ID number data field).

I have to use 2 different data structures because you can’t really sort strings alphabetically and have fast data access at the same time by using just a single hash function. After all, a hash function’s purpose is to avoid the long traversing part, to retrieve items fast. But I also need an ordered structure type so that I can sort the names when asked to print them.



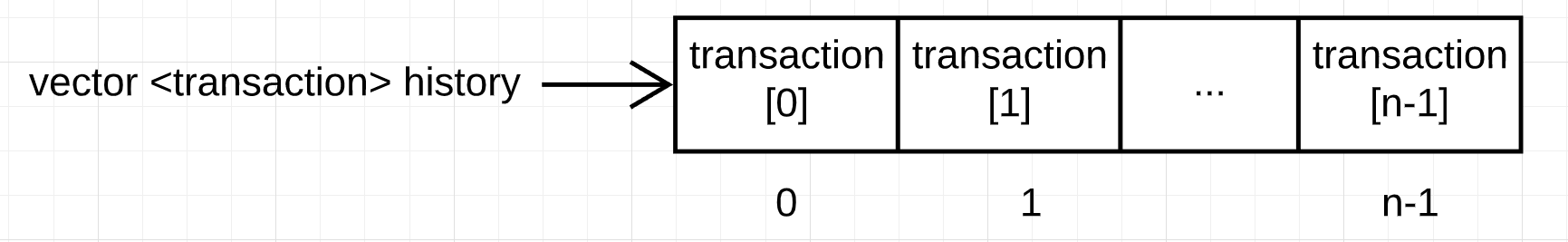
CustomerName BST:



1. **In the Customer class**

Each Customer has a vector that stores their personal list of transaction at the store: any items bought or sold. A “transaction” struct is defined as pair of transaction type (S,B) and transaction item (Coin, Comic, SportsCard).

Each Customer has a String name field and other operations to buy/sell/display items from the store.



1. **In Shop class**

Each char read in from the Items input file will be mapped to its corresponding Item type in a hash table. Each slot of the array is a dummy instance of the Item object so that during runtime, the factory method create() in the Item class will initialize with the correct child class Coin, Comic or Sports Card depending on the parameters passed in during initialization.

