Chase Howard
UW Bothell
CSS 502B: Olson
2021-02-24

Object Oriented Design: Collectibles Inventory

Overview

This program will automate inventory tracking for a local collectibles store. It will consist of the following objects:

- `Store`: Entity containing an `Inventory` and `History`
- `AVLTree`: Balance Binary Search Tree keeping record of `Customer` interactions with the store (buying and selling)
- `Customer`: Entity responsible for buying or selling an `Item` within the `Store`
- `Transaction:` Object containing `Collectable` and indication of buying or selling event tied to a `Customers` history
- `HashTable`: Hash Table containing a collection of `Item` bought or sold within the `Store`
- `Collectable`: Entity bought or sold at the `Store` including the following:
    - `Coin`: `Collectable` described by [type, year, grade (int)]
    - `ComicBook`: `Collectable` described by [year, grade(string), title, publisher]
    - `Sports Card`: `Collectable` described by [year, grade(string), player, manufacturer]

The main function will instantiate a Store object followed by allowing input of files for populating the contents of the inventory from a file, then input a list of customers from a separate file, and finally process the arbitrary sequence of commands from a third file. Example of what the client may look like in main:

```
int main(){
   create myStore; // create empty Store object
   myStore.inputFile(file);
   myStore.inputFile(file);
   myStore.inputFile(file);
   myStore.display() // print contents of inventory to ostream
   myStore.history() // print history for each customer in chron. order
   myStore.inputFile(file);
   myStore.inputFile(file);
   myStore.display();
}
```

Formatting of each input file will be correct; however, there is no guarantee that information within the file will be considered valid. Sample formatting of files is described below.

Inventory List:

- The data file for initialization of the inventory lists each item on a separate line with a character denoting the type of item, number in inventory, year, grade, type/title/player, publisher/manufacture (if necessary). Fields are separated by commas as follows:

```
M, 3, 2001, 65, Lincoln Cent
M, 10, 1913, 70, Liberty Nickel
C, 1, 1938, Mint, Superman, DC
C, 2, 2010, Excellent, X-Men, Marvel
Z, 4, 1986, Raging, Metallica, Master of Puppets
S, 9, 1989, Near Mint, Ken Griffey Jr., Upper Deck
S, 1, 1952, Very Good, Mickey Mantle, Topps
```

Customer Information:

- Customer information is stored in a similar file. Customers have a 3-digit ID number that uniquely identifies them:

```
001, Serena Williams
456, Keyser Soze
999, Pele
```

Command List:

- Your code will be tested using a third file containing the commands (S for Sell, B for Buy, D for Display, C for Customer, and H for History). Except for the Display and History commands, the second field is customer ID. For buy and sell commands, the third field is the item type and the remaining fields are the item. In the example below, "X" may indicate an invalid entry in the command list.

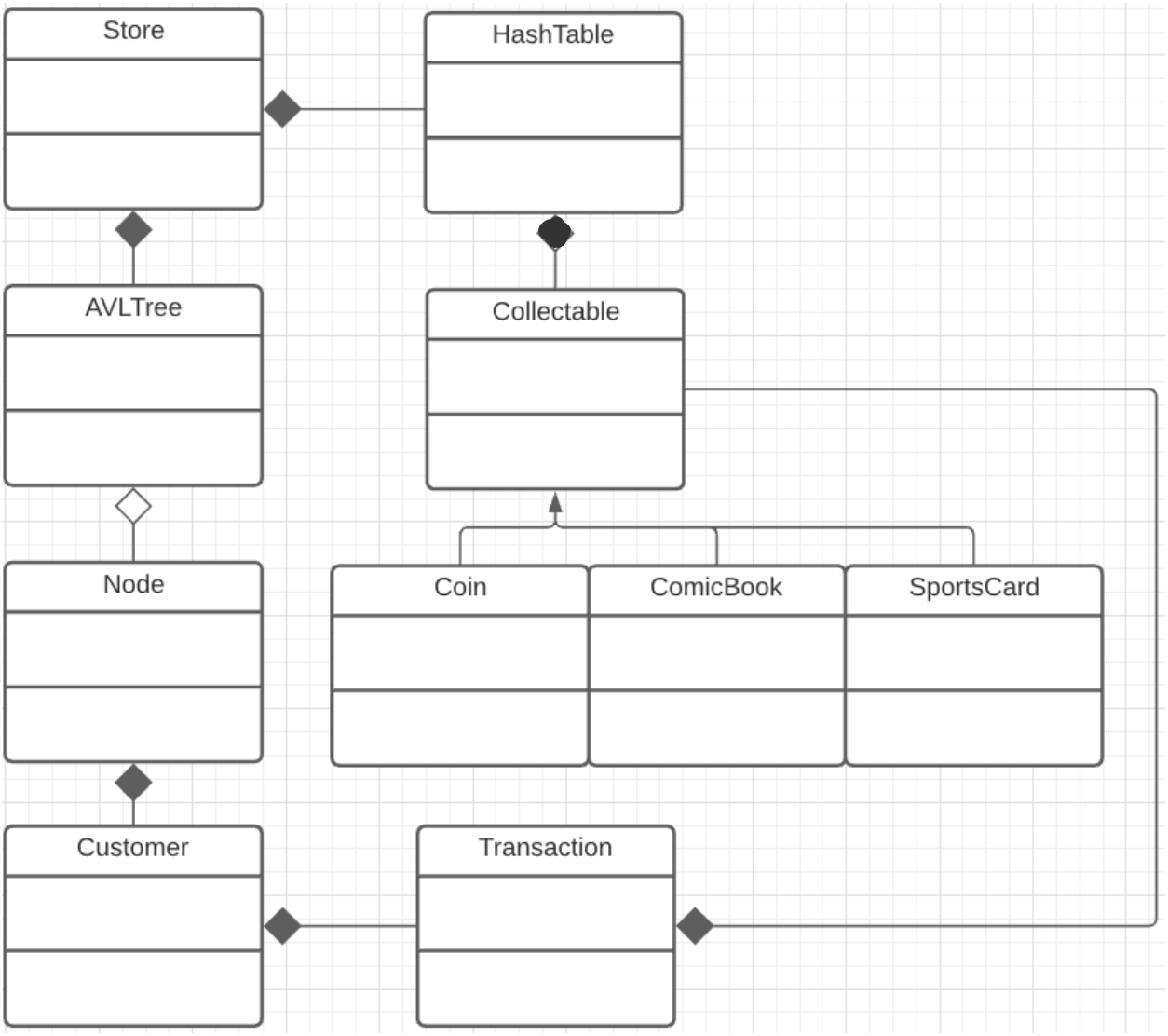
```
S, 001, S, 1989, Near Mint, Ken Griffey Jr., Upper Deck
B, 456, M, 1913, 70, Liberty Nickel
C, 999
D
X, 999, Z, 1986, Raging, Metallica, Master of Puppets
S, 000, Q, 2112, Gnarly, Windows, Microsoft
H
```

Note the inclusion of characters undefined ('D', 'X') would be considered invalid input. The program will be capable of taking care of common processes associated with an inventory as defined below:

- Sell: removes an item from the store Inventory
- Buy: adds an item to the store inventory
- Display: outputs the entire inventory of the store, including the number of each item in inventory. Output will be in increasing order with the following stipulations
    - Coins are sorted first by type, then year, then grade
    - Comics are sorted first by publisher, then title, then year, then grade
    - Sports cards are sorted by player, then year, then manufacturer, then grade

    - Coins then Comics then SportsCards
- Customer: outputs all the transactions for a customer (in chronological order), including the item
- History: outputs the history for every customer, with the customers in alphabetical order.

Class diagram:



Memory diagram: