Matthew Park

Coupon Generator Documentation

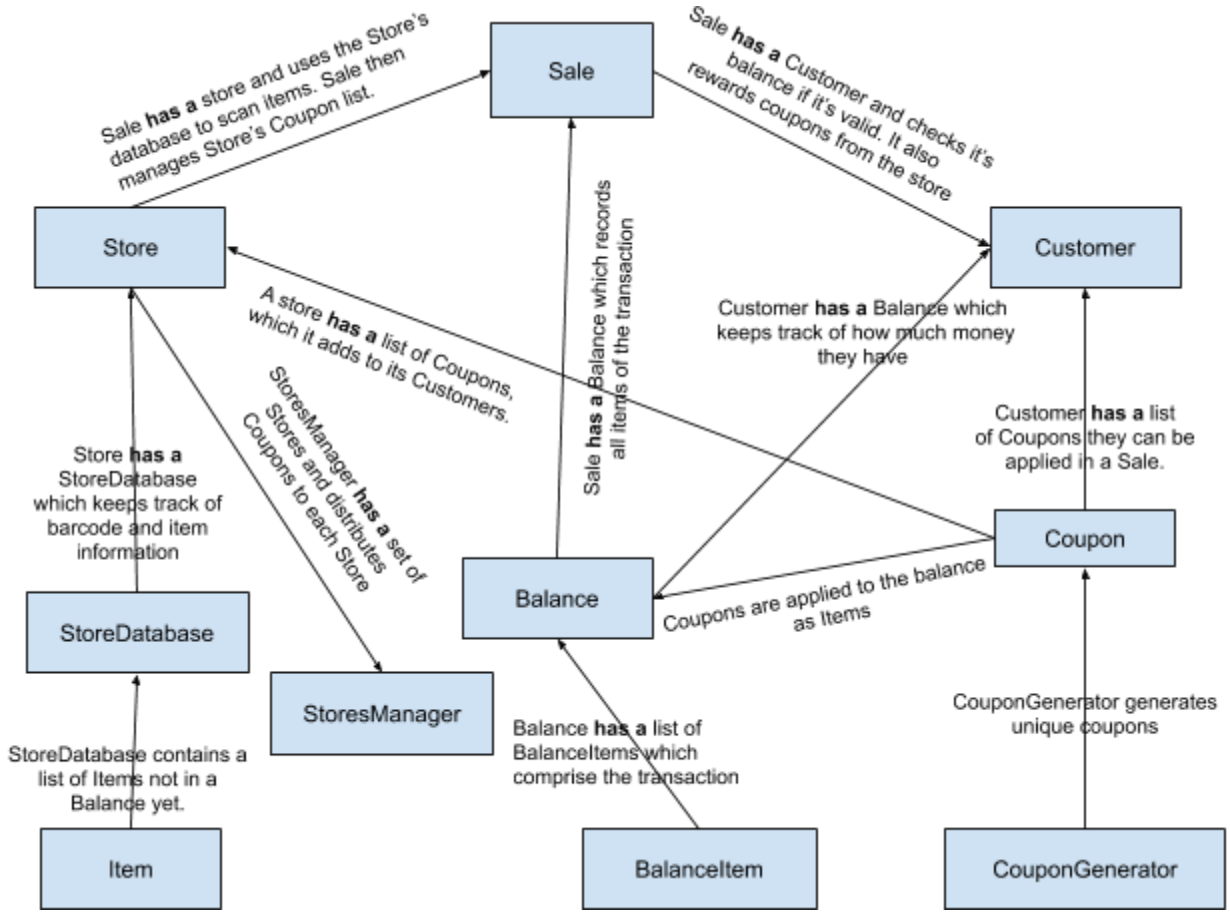
Class Summaries

Below is a quick summary for each class involved.

Sale	Sale contains all information of a transaction which includes the Customer, the Store, the StoreManager, and Balance. Sale is responsible for scanning items and processing and distributing coupons.
Store	Store contains a store's coupon inventory and store database. Store is in charge of managing its own coupons and rewarding coupons to customers.
StoresDatabase	StoresDatabase loads and stores all item information from a file.
StoresManager	StoresManager keeps track of all stores involved and distributes coupons to them.
Customer	Customer contains all information about a customer's name, balance, and coupons. A Customer may receive and use coupons in a transaction.
Coupon	Coupon stores a coupon's rule, unique code for identification, and whether or not its used.
Balance	Balance stores all items for one party of a transaction. Balance for a Customer typically stores the money they have, while Balance for a Sale typically stores the items a Customer wants to buy.
BalanceItem	BalanceItem is a storage object that contains the amount, category, description, and discounted price for an item in one's balance.
Item	Item is a storage object that stores information about an item that has not been sold yet.

Class Association

Below is a visual for how these classes interact during a transaction:



Example Walkthrough

Below is a walkthrough explanation of how each Sale is processed.

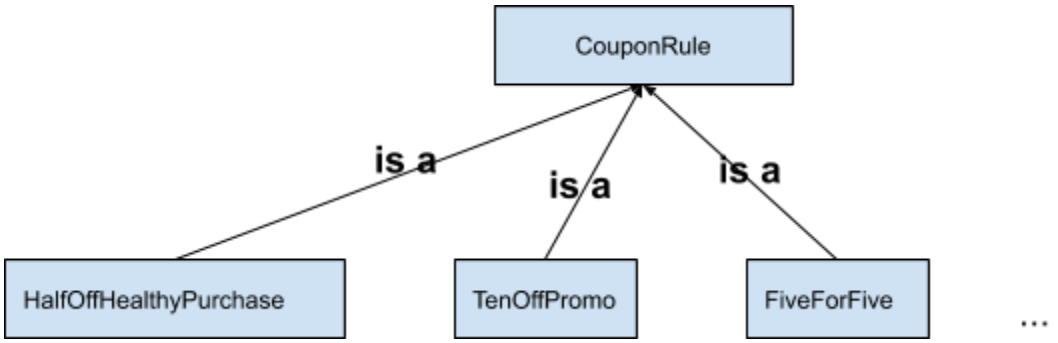
1. First, a Customer object, who wants to buy items, is needed.
2. The Customer **has a** Balance object. If the Customer has money, we will add balance to the Customer's Balance using add_balance().
3. A Store object is needed, which is the place the Customer buys items. Store also receives a StoreDatabase, which loads all barcode and item information from a file.
4. We must register the Store object to the StoresManager. This will also issue Coupons to the Store based on what is in the INITIAL_COUPON_STOCK.
5. We award Coupons to the Customer using reward_coupons(). Customer stores its Coupons in a list.
6. For every transaction, we create a Sale object. A Sale object **has** the Store and Customer.
7. We scan barcodes of the items the Customer wants to buy. We use sale.scan_item(barcode) to process this. Sale accesses the Store's StoreDatabase to find

information on the Item with the passed barcode. Sale **has a** Balance to keep track of all items the Customer wants to buy. All items are added into Sale's Balance as BalanceItems.

8. We use the Customer's coupons by calling sale.use_coupons(coupons). This iterates through the list of coupons and applies them if they are valid and unused.
9. We call sale.finish_sale() to finish the transaction.
    a. First, Sale checks if Customer's Balance is higher than the Sale's Balance. If not, the Customer does not have enough money, and we throw an Exception.
    b. Sale then subtracts the sale amount from the Customer's Balance.
    c. Sale groups the items in Sale's Balance together for a neater output.
    d. Sale adds Coupons to the Customer's Coupon list by calling reward_coupons().

CouponRule Inheritance

Let's discuss the Coupon class. A Coupon contains three pieces of information: the rule, if its used, and its unique code. A Coupon's code is generated uniquely by CouponGenerator. Coupon's rule is determined by a subclass of the abstract type CouponRule.



Subclasses must implement the static functions description(), can_use(), number_to_reward(), and apply_discount(). So far, there are three types of coupons: HalfOffHealthyPurchases, TenOffPromo, and FiveForFive. This **is-a** relationship is flexible in that new coupon types can be added without extensive modifications to different parts of the program.