**The Panda’s Next Gen POS Developer’s Manual**

**1. Introduction**

Next Gen POS was developed for an easy solution to typical retail scenarios. The system includes basic functionality for sale, rental, return, and user management. This manual is intended for developers that are familiar with Java and SQLite integrated applications. This manual is organized by the essential use cases, sale, rental, return, and user management.

**2. Core Functionalities**

* Sale (Transaction) - this function represents a customer bringing some quantity of items to a cashier, then the cashier enters each item’s ID and the POS will process the sale and calculate a tax, subtotal, and grand total
* Rental - customers can rent items that are marked as rentable in the DB, they are asked for their desired rental period, and the necessary DB changes are made and recorded. Rentals are marked on a customer receipt
  + Customers are charged a rental deposit to guarantee the return of items
* Return - users can return items, these normal(non rentable) items are placed in a separate DB table; rentable items that are returned have their original cart’s date checked against the current date at which the customer is trying to return the items, if these items are past due they are charged a fee of .625% of the rental price.
  + when returning rentable items that ARE on time, the rental deposit is calculated per item and returned to the customer via cash/credit depending on their original payment method as indicated with a receipt
* User Management - There is a primary login procedure in which an employee(Cashier/Manager) will log in, and grant them access to the above mentioned functionalities. Managers have special privilege to add, delete users from the User DB table.

**3. Getting Started**

**3.1 System Requirements**

* Java 6+
* SQLite
* Java and SQL compatible IDE
* UNIX based command line

**3.2 Components**

|  |  |
| --- | --- |
| Primary File | DB Files |
| Transaction.java | data.db |
| Rental.java | SQLInterface.java |
| Return.java | ReceiptManager.java |
| Register.java |  |
| Item.java |  |
| Cart.java |  |
| Cashier.java |  |
| Manager.java |  |
| Login.java |  |
| Receipt.java |  |
| TaxCalculator.java |  |

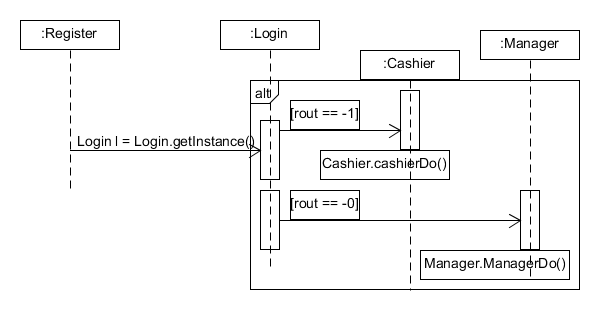
**3.3 Third Party Components**

SQLite component is used for DB management. SQLInterface.java and ReceiptManager.java are the primary components of the system that implement and call SQLite commands. From these classes, calls are made to generate SQL commands that will then interact with the SQLite interface to generate/modify the appropriate entries.

**4. Developer’s Guide**

**4.1 - About the POS System**

* Register.java contains the main method, upon starting up the system, one will be prompted with a menu from here.



* Sequence Diagram illustrates login procedure to access the core components of the POS system (Cashier/Manager functions)

**4.1.1.1 - Creating a new Use Case**

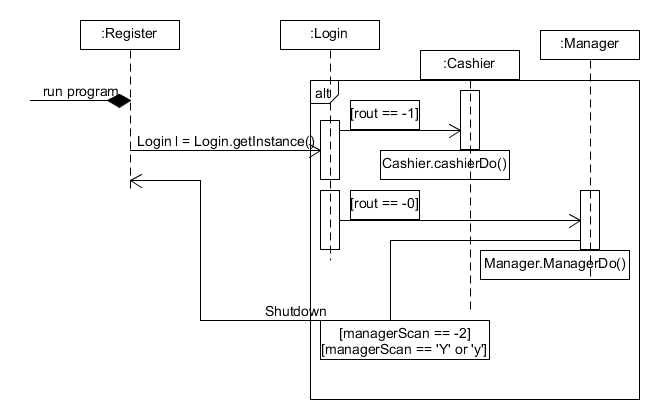
* To create a new use case, create a new class that would extend the Register class so the login procedure could still apply.
* Add necessary functions to the proper user (Cashier/Manager/Both) so that the proper permissions still apply for the new use case

**4.2 - System Startup and Shutdown**

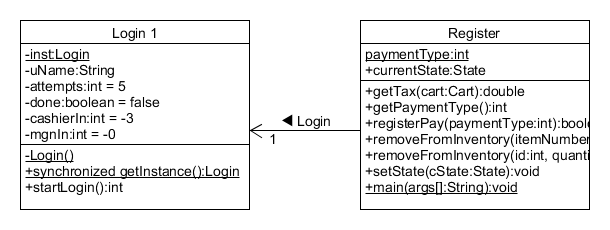
**4.2.1 - Design and Implementation of System Startup and Shutdown**

* System Startup is represented by running the program and executing main which puts the user into the ‘main menu’ awaiting a login
* Shutdown can only be performed as a manager, and will prompt the user for verification that they wish to shutdown the system and therefore exit the program
* At the point of shutdown, there is no concern to updating or saving the DB as it is currently locally hosted on a personal machine. In addition any operations done related to the DB immediately updates the DB.

**4.2.1.1 - Architecture View**



**4.2.1.2 - Class Diagrams**



**4.2.1.3 - Data View**



**4.2.2 - Customize your own Startup and Shutdown**

* One can attach the database system to a server rather than having it locally hosted per machine. If doing this, then one must ensure connections to the server are closed to minimize risk of data loss. A simple function could be added to the Manager class, that upon notice of wanting to shutdown to close all network connections.

**4.2.2.1 - Customizing Startup/Shutdown Examples**

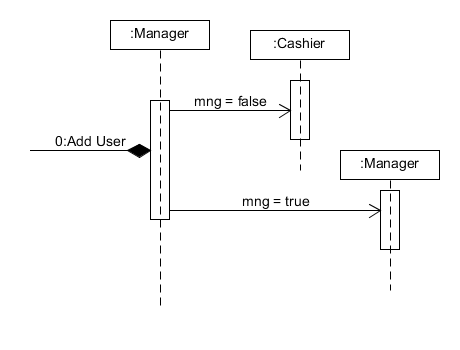
* Ex1 - You have integrated a third party payment authorization service with the system, prior to fully starting up the system, you must ensure the proper connections are established and when shutting down you must ensure these same connections have been closed once all data has been properly stored/transferred.

**4.3 - User Management**

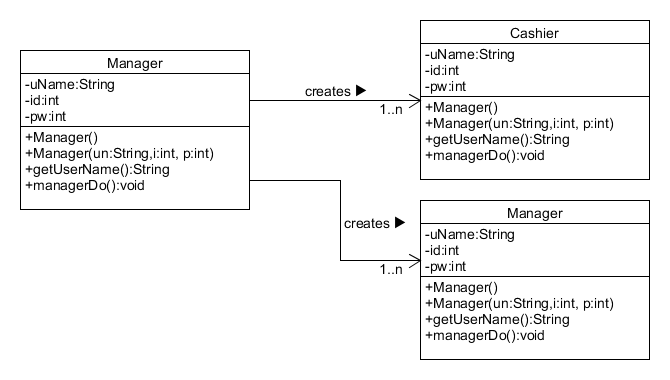
**4.3.1 - Design and Implementation of User Management**

* User management is represented by editing user information in the system.
* System acquires an administrative level authentication to to edit the user information in the database.
* System updates its database when the user information is successfully edited.
* System manager should log out to go back to regular level of security.

**4.3.1.1 - Architecture View**



**4.3.1.2 - Class Diagrams**



**4.3.1.3 - Data View**



**4.3.2 - Customizing User Management**

One can customize what privileges different users have, or

create different user archetypes in order to simplify the process.

**4.3.2.1 - Customizing User Management Examples**

Currently two levels of users exist: administrators and cashiers. One could

create a third type say, senior cashier, which can startup and shutdown the

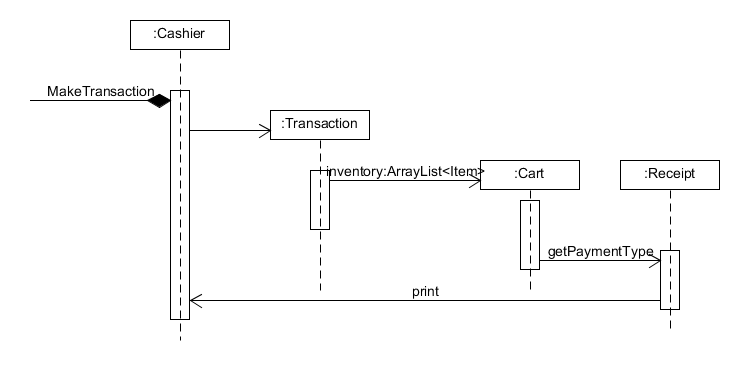
system, but not add users.

**4.4 - Process Sale**

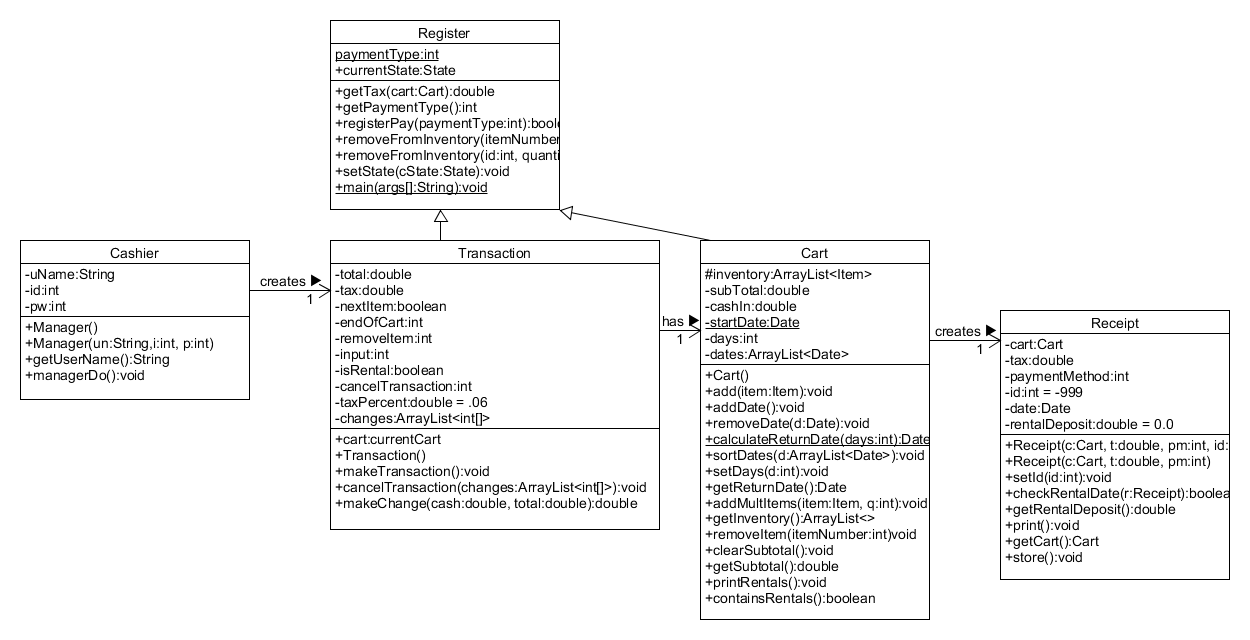
**4.4.1 - Design and Implementation of Process Sale**

* Process Sale is represented by managing the sales in retail stores.
* Process sale should retrieve the name and price of a product from the backend catalog and interact with inventory system to update the stock amount of a product.
* Process Sale allows the customer to pay in either cash or credit card(Visa).

**4.4.1.1 - Architecture View**



**4.4.1.2 - Class Diagrams**



**4.4.1.3 - Data View**



**4.4.2 - Customizing Process Sale**

**-**One can customize multiple parts of process sale. Anything from calculating

taxes to printing receipts. You can change the way multiple items are purchased

(whether you scan one item 3 times to buy three, or whether you can it once and

enter quantity = 3).

-The customer can use the coupons for a better price

**4.4.2.1 - Customizing Process Sale Examples**

-Example 1: Your business only sells to non-profit companies. As such they

never pay sales tax, so you can remove the tax calculator and edit the formatting

of the receipts to remove the tax fields.

-Example 2: Your business only sells unique items (like an antique store). So

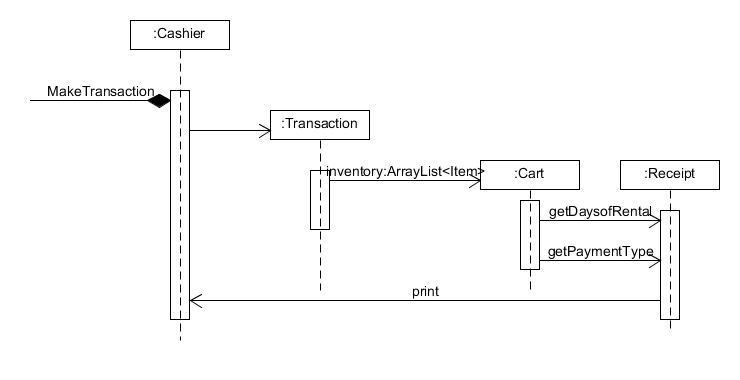
you never will have multiple of any item. So you can edit the process sale so that it doesn’t ask for quantity each time you scan an item.

**4.5 - Process Rental**

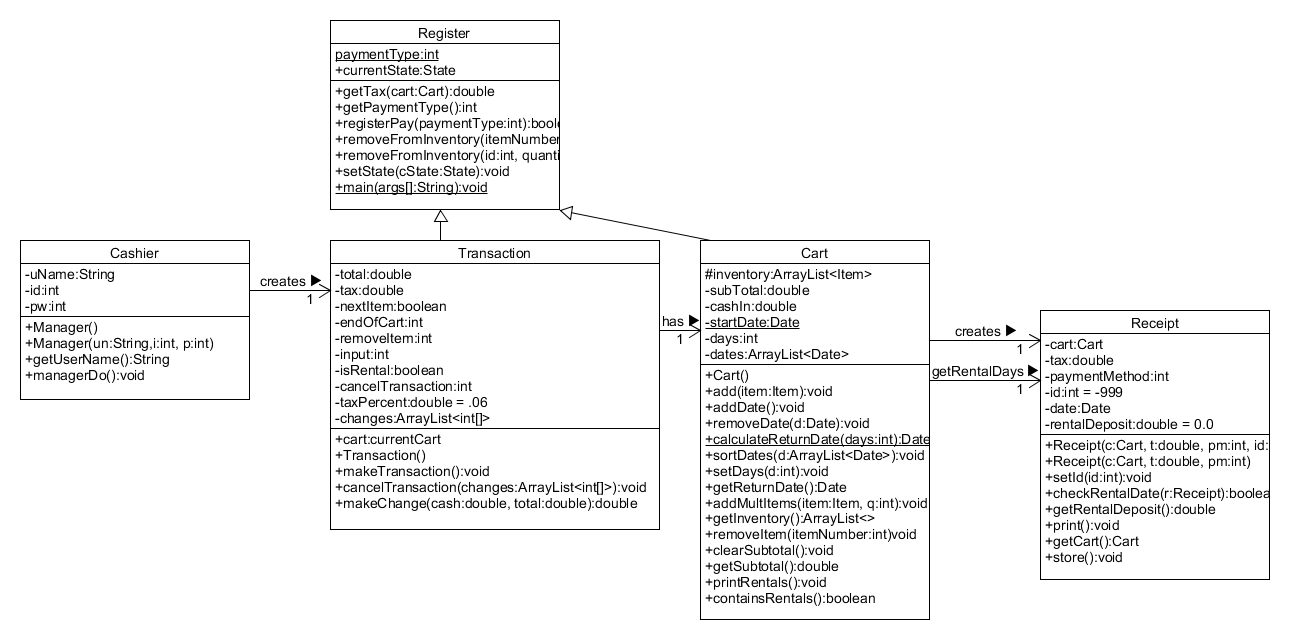
**4.5.1 - Design and Implementation of Process Rental**

* Process Rental is represented by renting goods for certain period of time, and updating inventory.
* Unless customer has personal information in the system, he/she has to provide information that store requires for new customer registration.
* System updates its inventory in both case when the product is rented and when the product is returned from renting.

**4.5.1.1 - Architecture View**



**4.5.1.2 - Class Diagrams**



**4.5.1.3 - Data View**



**4.5.2 - Customizing Process Rental**

Process rental can also be customized according to the consumers needs. The

length of rental periods can be set to a standard length or can be adjusted for

every item. The cost of renting and item can be whatever the customer wants it

to be. Items may or may not be returned early for a partial refund, and many of customizations are possible.

**4.5.2.1 - Customizing Process Rental Examples**

Suppose your business is a Ski rental shop. You rent out various skiing gear, but

only for a weekend at a time. In this scenario you don't want to have to set the

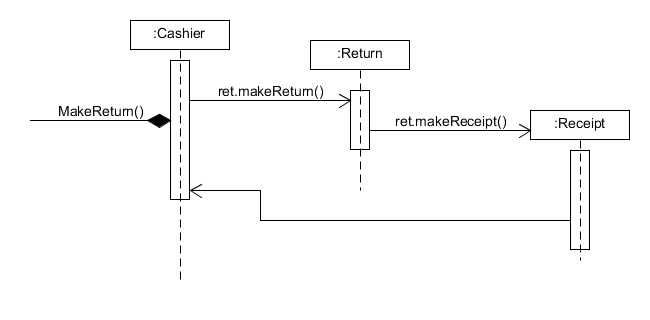
duration of each rental, since it's always the same. So the rental process can be customized to suit that.

**4.6 - Process Return**

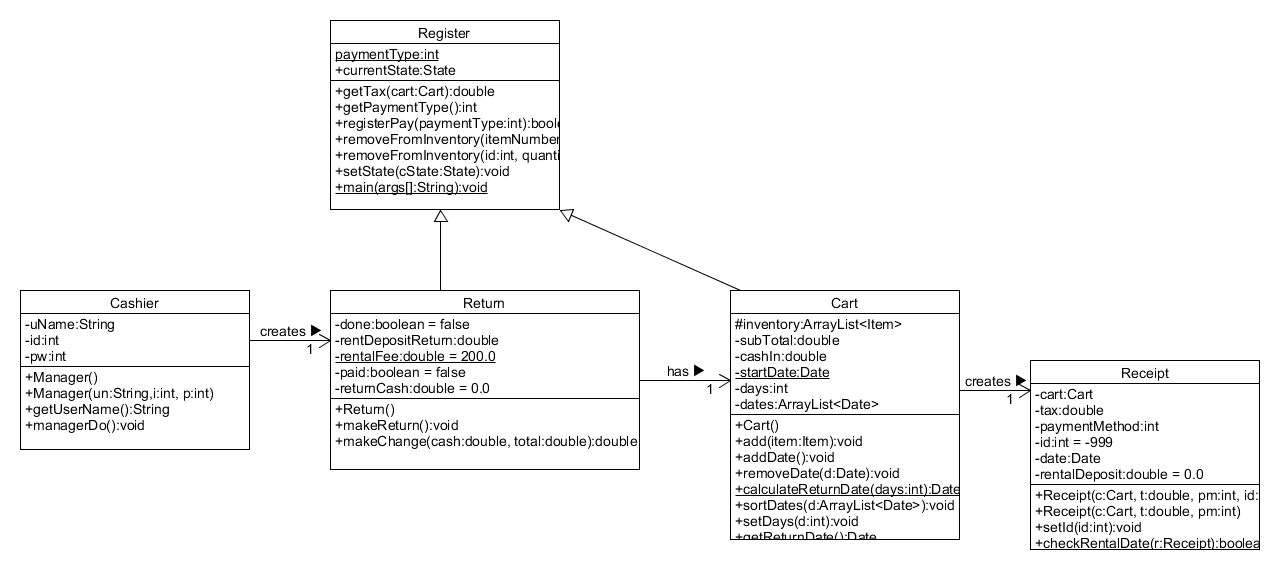
**4.6.1 - Design and Implementation of Process Return**

* Process return was designed to handle a normal return and returning a rental. When the item returned is a normal item, the item is placed in a separate DB table. Then the customer is refunded either change or credit depending on their original payment method.
* If the item is a rented item, the return date of the item is compared to the date on the receipt. If it is a valid date, the item is returned and the rental deposit taken from the customer is returned to them per item. However, if the rental is past due a fee is charged to the customer and prompts them for payment.

**4.6.1.1 - Architecture View**



**4.6.1.2 - Class Diagrams**



**4.6.1.3 - Data View**



**4.6.2 - Customizing Process Return**

Process return can be customized in a variety of ways as well. It can be modified

to change how much money gets returned upon return, or what period of time an

item can be returned within, or potentially completely removed so that no returns

are possible.

**4.6.2.1 - Customizing Process Return Examples**

Suppose your company is a restaurant. Returns are never possible with food, so

the return process can be completely removed because it is unnecessary.

**4.7 Customizing the Database**

* To expand the database, first be sure to have the latest version of SQLite installed on the respective machine. Note that database entries are local hosted to a single machine. To expand this, be sure to set up a SQLite server and open connections via an additional class.
* Adding new entries - Create new methods following similar patterns of creating SQL commands and passing those to SQLite application. This will allow one to make new entries for products, users, and/or receipts.

**4.8 Vocabulary**

* DB - database, stores the item information, receipts, returns, and user information
* POS - point of sale system - the ‘register’ that will carry out sale, returns, rental, it is the main element that provides functionality to a user