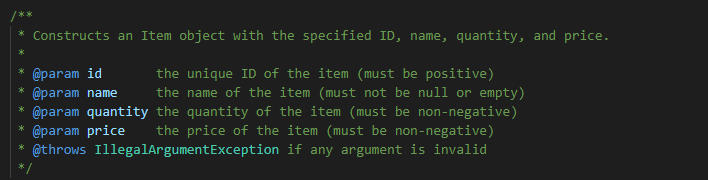
There is some working of the whole program:

1. The DateTime class manages as the name says Date and Time. It Has multiple attributes such as minute, hour, day name etc. The most important function in the class is the setCurrentTime() which will automatically set the current Date and Time. Class also has all the other basic things such as constructor, Getter and Setters etc.
2. Item Store the basic things related to an item such as ID, Name, Quantity and Price. It implements an interface. The attributes have some restrictions such as id cannot be less than 1 etc. All the restriction are in the class file mention with detail.



1. **The order class is somewhat more unique to the other classes. It has its own attributes such as OrderNo, Customer etc. But it also inherited from DateTime Class because each order has a TimeStamp of Date and Time. Also, each order can have multiple items in the cart. So, I also created an arraylist inside the order class to have multiple items in the arraylist. The Class Also have multiple functions such as getTotalPrice() which calculate the Total Price that customer have to pay in that order. Also, Constructor, getter and Setters are in there.**
2. One of the Most important classes in the whole program is the FileHandler which contains some static functions to work around files. They are used to store and read data from file. Each Function is unique and store different type of data in the filename specified by the user. There are 3 types of files which are Items.csv (Store the Data of Items in the Store), Orders.csv (Store the Order Data such as Customer Name etc.) and Last OrderItems.csv (items that were part of an order identified by a order number).
3. The OrderManager and ItemManager classes have Some What Same type of functionality. They are used to Search, Update or Remove things from the order or item class arraylist. They are also static which mean that we can’t have to create an instance of these classes.
4. The Main program just binds all the things together in a more user-friendly interface. It Display Main or Submenu. Stores the Data and Also perform tasks on user selection.

App.java:

This Java program is designed to simulate an ordering system for a store. It contains the main method and handles the overall flow of ordering process implementations. The user is presented with a main menu that has three options: display orders, display items, or save and close. If the user chooses to display orders, they are presented with an order menu with four options: add a new order, remove an order, view an order's details, or display the last 10 orders.

The program uses two ArrayLists, orderList and itemList, to store all program data. The program begins by attempting to read data from two files, "Orders.csv" and "Items.csv", using the FileHandler class. If the files are not present or an error occurs while reading the files, the program initializes the ArrayLists with zero data.

The mainMenu() method displays the main menu and returns the user's selection as an integer. It uses a while loop and a Scanner object to ensure that the user enters a valid option.

The main() method is the starting point of the program. It first attempts to get all the data from the system files by calling FileHandler.readItemsDataFromCSV() and FileHandler.readOrdersDataFromCSV(). If the files are not present, the ArrayLists are initialized with zero data.

The program then enters a while loop that displays the main menu and executes the appropriate function based on the user's selection. If the user chooses to display orders, the program enters another while loop that displays the order menu and executes the appropriate function based on the user's selection.

If the user selects "add a new order," the program prompts the user for the customer's name and a list of items. It then calls the ItemManager.itemSearch() method to search for the desired item and asks the user for the quantity. If the requested quantity is available in stock, the program creates a new Item object and adds it to the user's cart. If the requested quantity is not available, the program informs the user and continues with the loop. The user is then prompted to add another item or finish the order. If the user selects "remove an order," the program prompts the user to search for the order by order number and removes the order from the orderList if it exists. If the user selects "view an order's details," the program prompts the user to search for the order by order number and displays the details of the order if it exists. If the user selects "display the last 10 orders," the program displays the last 10 orders in the orderList.

Overall, this program provides a simple ordering system with basic functionality, such as adding, removing, and viewing orders, and reading and writing data to CSV files. It could be extended to include more features, such as customer information, payment processing, and data visualization.

FileHandler.java:  
  
This Java code provides a FileHandler class that includes utility methods to write data to CSV files and clear file data. The class includes the following methods:

writeItemsDataToCSV: This method writes a list of Item objects to a CSV file with the specified file name. If the file already exists, all the previous data will be deleted.

clearFileData: This method clears the contents of a file with the specified file name.

readItemsDataFromCSV: This method reads the contents of a CSV file with the specified file name and returns an ArrayList of Item objects.

writeOrderedItemsDataToCSV: This method writes a list of OrderedItem objects to a CSV file with the specified file name. If the file already exists, previous data will be removed.

readOrderNoItemsDataFromCSV: This method reads the contents of a CSV file with the specified order number and returns the data of that order number in a list.

writeOrdersDataToCSV: This method writes a list of Order objects to a CSV file with the specified file name. If the file already exists, previous data will be removed.

readOrdersDataFromCSV: This method reads the contents of a CSV file with the specified file name and returns an ArrayList of Order objects

In the writeOrdersDataToCSV method, a new FileWriter is created and opened with the specified fileName. The method then writes the header row to the file using the write method of the FileWriter object. The header row consists of the column names for the orders data: orderNo,orderDate,orderTime,totalPrice.

The method then writes each Order object in the orders ArrayList to the file. For each Order, the method retrieves its order number, date, time, and total price using its getter methods. It then writes these values to the file, separated by commas and followed by a newline character.

After all the Order objects have been written to the file, the FileWriter object is closed using the close method.

Overall, this method writes the contents of an ArrayList of Order objects to a CSV file with the specified file name. If the file already exists, all previous data will be deleted before writing the new data.

DateTimeInterface.java

Graphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generated

Datetime.java: