Project: SP2019\_PROJECT\_Valeriano

(word wrap alt+shift+Y)

**Summary of Project**

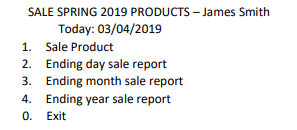
An application that lets employees print a receipt on screen for each sale transaction.

It also allows sales reports to be printed out at the end of a day, month, or year.

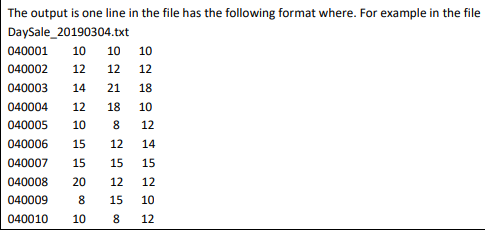
**Driver class: SaleSP19Report\_Valeriano**

Variables:

* Current date
* Displays a main menu for the user



* Switch on a loop and call methods from data type class
* These methods are located in the driver class:
  + **Sale Product (a transaction is made here)**
    - User is asked for input for the following:
      * How many SP191s to buy?
      * How many SP192s to buy?
      * How many SP193s to buy?
      * How much did the customer pay?
    - The number of units bought is passed to a constructor for a new instance of data class and its this class that calls all the methods
    - Make transaction number based off date, call printReceipt method
    - Call method to write to the daily output file (for each time Sale Product is called and it’s still the same day, increase the transaction number by 1, otherwise reset it using the new date. Turn it into a string)



* + **Ending Day Report (to be run at the end of a day. Gets info from day file made in Sale Product. The day itself is added to a new month file)**
    - Ask users what day to run the report in using this format: mm/dd/yyyy
    - Tokenize string into mm, dd, yyyy and create or look for a file called **daySale\_yyyymmdd.txt**
    - Open that file for reading (use a scanner)
    - Call TokenizeLine for each line and add up # of units sold
    - Close the file
    - With this information, **create a new ProductSP19\_Valeriano object** and pass in the # of each unit sold in constructor
    - Call object’s corresponding DisplayReport method and pass in date from earlier
    - Create a new month file (if it doesn’t exist already) called **monthSale\_yyyymm.txt** and write one line to the file like below:



* + - * Column 1 is the day l
      * Then each column after is the # of respective SP191, 192, and 193 units sold
  + **Ending Month Report (to be called at the end of a month)**
    - Ask users what month to run the report in using this format: mm/yyyy
    - Tokenize input string into mm and yyyy then create or look for a file called **monthSale\_yyyymm.txt** (might be created already in Ending Day Report)
    - Open that file for reading (use a scanner)
    - Call TokenizeLine for each line and add up # of units sold
    - Close the file
    - With this information, **create a new ProductSP19\_Valeriano object** and pass in the # of each unit sold in constructor
    - Call object’s corresponding DisplayReport method
    - Create a new year file If it doesn’t exist already called **yearSale\_yy.txt** and write one line to it like below:



* + - Column 1 is the month and the other 3 are for unit #s sold
  + **Ending Year Report (run from the end of a year)**
    - Ask user for which year to run the report in format of yyyy
    - Create or find file **yearSale\_yyyy.txt** then open it for reading (scanner)
    - Call TokenizeLine for each line and add up # of units sold
    - Close the file
    - With this information, **create a new ProductSP19\_Valeriano object** and pass in the # of each unit sold in constructor
    - Call object’s corresponding DisplayReport method
  + **TokenizeString (splits a string according to a delimiter and returns an array of tokens)**
    - Tokenize each line from the file to get…
      * # of SP191 units
      * # of SP192 units
      * # of SP193 units
    - Return an array?

---------------------------------------------------------------------------------------------------

**Data type class: ProductSP19\_Valeriano**

* Holds information on…
  + # of SP191 units with price of 12.99 each
  + # of SP192 units for 14.99 each
  + # of SP193 units for 15.99 each
* Has methods for…
  + **Constructor**
    - Take in # of units when creating an object of class
  + C**alculating earnings from the SP191s sold**
    - Sale model SP191 = unit price of SP191 \* # of SP191s sold
  + **Calculating earnings from the SP192s sold**
    - Sale model SP192 = unit price of SP192 \* # of SP192s sold
  + **Calculating earnings from the SP193s sold**
    - Sale model SP193 = unit price of SP193 \* # of SP193s sold
  + **Calculate sub total**
    - Sales of 191, 192, and 193 combined
  + **Calculate Tax**
    - Subtotal \* 8.25% (.0825?)
  + **Calculate total**
    - Subtotal + tax
  + **Calculate Balance**
    - The amount the user decides to pay – total
    - Where does the user put in the amount they want to pay?
  + **Write a line to an output file of day**
    - Pass in a transaction number and a string for the file name
    - Find and open the file using the name passed in for writing
    - A line is split into 4 columns for…
      * Transaction # (transaction # passed in is placed here)
      * # of SP191 units sold
      * # of SP192 units sold
      * # of SP193 units sold
    - Example:



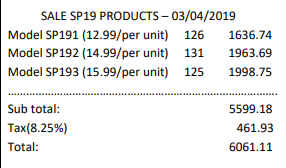
* + **Write a line of output to a file for a month (tracks # of units sold each day in a month)**
    - 4 columns for…
      * The current day? (from 1 to 31 for March)
      * # of SP191 units sold
      * # of SP192 units sold
      * # of SP193 units sold
    - Example for march 4th:

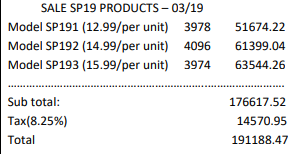


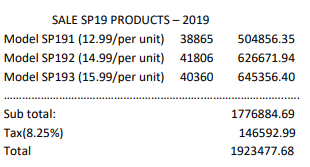
* + **Write line to an output file for a year (tracks # of units sold each month in a year)**
    - 4 columns for…
      * The month (1 to 12)
      * # of SP191 units sold
      * # of SP192 units sold
      * # of SP193 units sold
    - Example for march:



* + **DisplayReport methods**
    - Outputs one of the following:







* + **Printing out a receipt**
    - 