INTERNATIONAL BACCALAUREATE

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Prepa Tec Santa Catarina

Computer Science SL

Ferretería

Written by

Name: Mauro Amarante

Candidate Number: dvs580

School: Tecnologico de Monterrey-Campus Santa Catarina

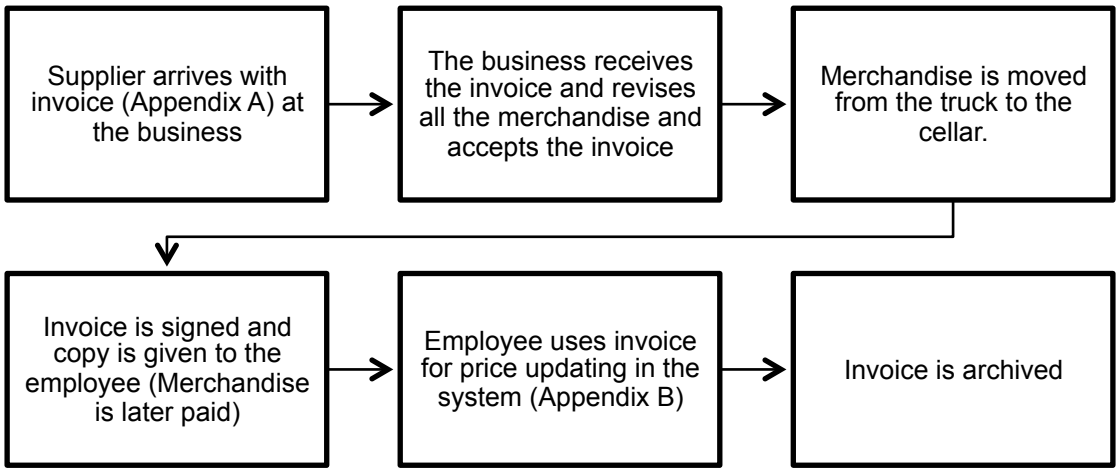
School Number: 002267

Table of Contents:

A1. Analysis of the Problem:

My client owns a local hardware store that offers very diverse merchandise that could be used for home use, industrial use, construction use, etc. From my client's diverse merchandise some of the products that he offers are screws, hammers, pipes, cement, sandpaper, light bulbs, electric utilities, paint, etc. My client has had some problems for a while now at his store with their organizational system for their merchandise. Currently my client is using a system-based organization that has been very useful for him in the past years, but has brought some problems that are affecting his business. This previously mentioned system (System Screenshot Appendix B) is very simple software that was previously bought and currently used by my client; he also mentioned "it has a simple but complicated interface for new or even current employees since it lacks friendliness for the user". Mainly these problems with the system are causing losses in profits and merchandise shortages that only damage the business and disrupt the compliance of their main objectives as a business (full problem description later in the text). My client's main objectives as a business are to never have a shortage of merchandise to avoid unhappy customers and to always have updated selling prices of their products to avoid loses in profits.

The problem with my client's business as seen by him is "stressful"; the losses in profits and some shortages on merchandise always makes it hard for my client to keep him happy and his customers at all times. First the losses in profit happen some days when prices require an update or, as called by my client, "price updating" is required. Sometimes prices need to be updated to avoid a loss in profit, for example, cooper is a raw material that is sold by my client in diverse shapes or forms as required by the client. In the case of copper if it has a rise in its cost, prices must be changed, but copper is not the only raw material that changes its price daily. There are many raw materials that are used in my client's products that are sold at the business that constantly change their cost. Therefore there is a daily process that is being done at the business due to this problem; a better description follows:



Here is a full description of each of the previously mentioned steps:

Event	Description
Supplier arrives with invoice (Appendix A) at the business	Usually at 8:00 AM in the morning merchandise arrives at the business from one or several suppliers.
The business receives the invoice and revises all the merchandise and accepts the invoice	When the merchandise arrives my client's employees need to check if all the merchandise that was ordered arrived in good state and that it was complete, after this the employees accept the invoice.
Merchandise is moved from the truck to the cellar.	Employees begin to move all the merchandise into the cellar behind the business.
Invoice is signed and copy is given to the employee (Merchandise is later paid)	When merchandise has being moved, my client signs the invoice and the copy is given to employee for him to proceed with the price updating. Merchandise is not paid at the moment; it is later paid through bank payment using the invoice.
Employee uses invoice for price updating in the system (Appendix B)	After receiving the copy of the invoice, an employee uses the invoice for the price updating by inputting he following data into the system:

	• Invoice Number
	• Date
	• Supplier Number
	• For every product:
	  ○ Quantity
	  ○ Fabrication Number
	  ○ Product's Description
	  ○ Product's Cost

Invoice is archived	When price updating is finished, invoice is archived in folders.

The information or data required for the system to work is the following:

Name	Description	Example (based on invoice in Appendix A)
Invoice Number	Identification number for every invoice that the business receives.	F39015
Date	Date of merchandise arrival and price change	10/01/2012
Supplier Number	Number that identifies the supplier (printed list with identification numbers for suppliers in the business).	07
Quantity	Number of units that were received from the supplier.	3
Fabrication Number	Number given by the business for identification (product's id number).	7501206611654
Product's Description	(Automatic): Product description that was previously saved in the system.	R.T. 125 x 50 x 15.9 MM. IC4BPE SHELLAC VERDE CLASICA

		PIEDRA
Product's Cost	Cost that is printed in the invoice.	83.48

During this whole process of price updating there have been issues that my client has encountered many times already in the past and that is that the date must be written every day, which then is written wrong by some of his employees that ten creates a lot of difficulties when wanting to check some data with saved paperwork or invoices. Also product's price has to be written every single time and employees input the prices wrong. Additionally when description is displayed it may be confusing since some descriptions are very similar between some products like screws since the only difference could be its diameter and mistakes are made sometimes too.

So there are lots of issues as seen by my client and he has previously tried to solve some of this issues by trying to make employees revise every thing that is written into the system twice, he told them to be very careful, he got them a calendar or the date writing, he also had two employees writing the data for them to check each others work but mistakes kept coming still since this is also a daily job. So one of his main problems is to reduce the human error caused by his employees.

Another main problem that my client has many times dealt with is merchandise shortage. At my client's business they do not have a system that reports if there is about to be a shortage or if they are completely out of stock in some products so usually when there is a shortage of any product that an employee "notices", they immediately report it so an order is placed for merchandise to arrive next week from the suppliers. Previously this was not a very big issue when all the data was managed in papers years ago since reports were received every day and the merchandise that was sold was removed from the books and if a product was coming to a shortage an order was made, but when the previously explained system was placed, it did not made the counting automatically every time a product is sold, but he needs to know when a shortage is about to happen. So the only way he has tried to solve this issue is by telling his employees to be careful with shortages and to report every time a product is about to come to a shortage which has worked but not as efficiently as he liked it would. Also he has tried to made an estimate of sells that are coming up and he would order in advance since he already knows the business works pretty well at every time in the year but having to much stock could also be expensive for my client.

So a possible solution could be a printed list for making a revision during the day for shortages revision if needed.

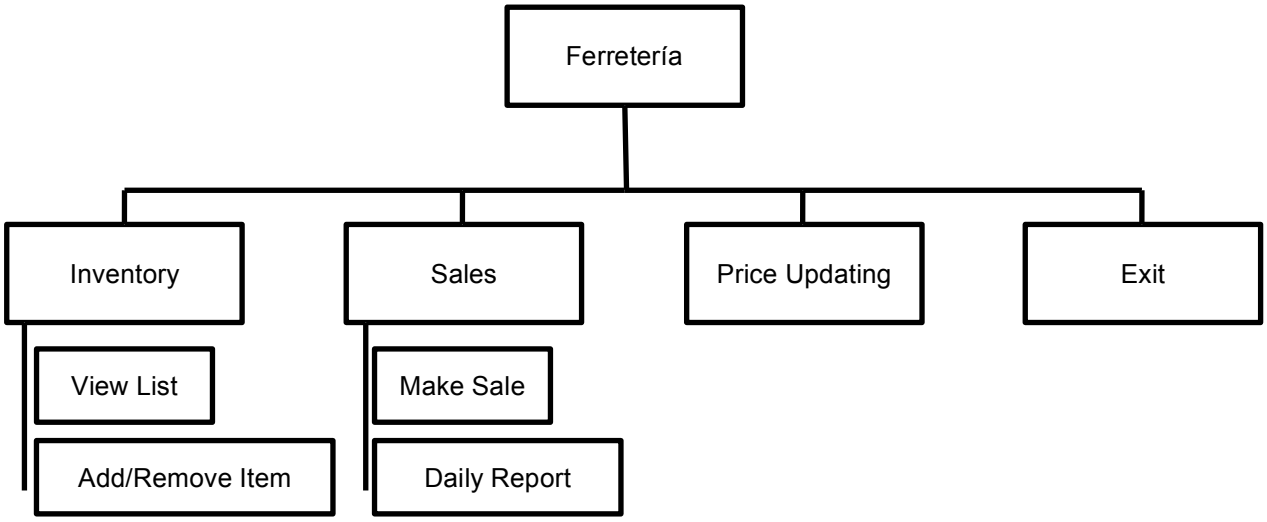
Also in the client's business obviously they sell their products. The current system that is implemented in the business is somewhat inefficient since it is complicated for the user to use in a fast way. As the client said, "It's frustrating to see many clients wandering in the store waiting for their turn". So then he decided to buy a numbered ticket roll where the clients would take a ticket and wait for their turn. So this make some change but not that much since the system still is not helping at all. Then the client said that even sometimes the employees mess up the sales and end up selling something way more cheaper and ten the business made a loss. Also the client noticed that most of his customers wont take the receipts and he thinks that is a waste of paper and of money too. This finally demonstrates that there are many issues in the store and trying to fix them is the client's biggest desire.

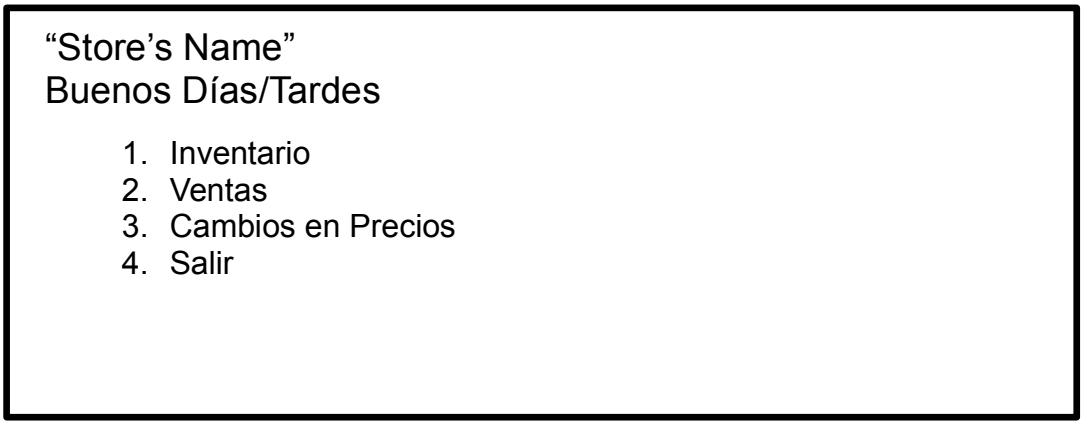
A2. Criteria for Success:

- Improve price-updating process:
    - Implement log system to ensure a database for possible revision of this process if required by the user.
    - Optimize the process by making it easier for the user to complete by the use of simple questions to complete.
    - Implement automatic date inputting for eliminating the possibility of user mistyping.
- User Friendly:
    - Easy to add, remove, make lists and make price updates with simple path to follow to accomplish a certain task.
    - If an error is present, the user will be notified about his mistake so it can be corrected.
    - Avoid mistyping of prices to evade the loss of profits in the business by making the user input the product's price twice.
- Inventory:
    - Implement shortage system section where the user will be notified if any product is currently out of stock.
    - Implement list printing method where the user can choose the group he wants the list of.
    - Give the user the capability to add or remove any desired product if there's a need to include a new one or remove one.
- Selling system:
    - Optimize selling system:
        - Easier use for improving the efficiency while making a sale to avoid unhappy waiting customers.
    - Include a full description of products when making sale so the user can be sure that the desired product is being sold.
    - Implement receipt creation system where the client can decide if he/she will like to have the receipt printed when products are paid at cashier.
    - Implement reports system:

- Give the user the capability to create a report that informs the user about the sales of the day.
- Give the user the capability to create a report that informs the user about the sales of the week.

A3. Prototype Solution

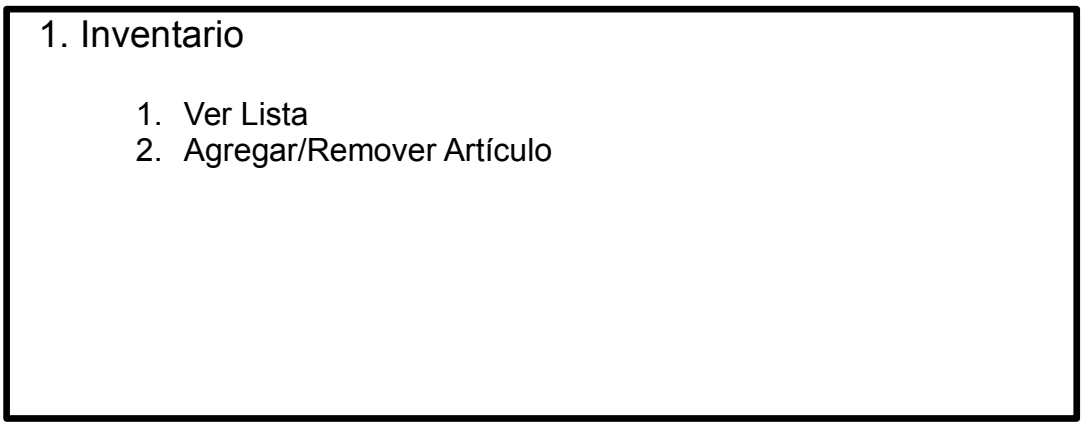




This is the opening menu for the program. It is written in spanish since my client's main language is spanish.

Feedback: *"Me parece perfecto, es lo que necesito."*

Translation: "*It seems perfect, just what I need."*



This is the menu for the inventory section.

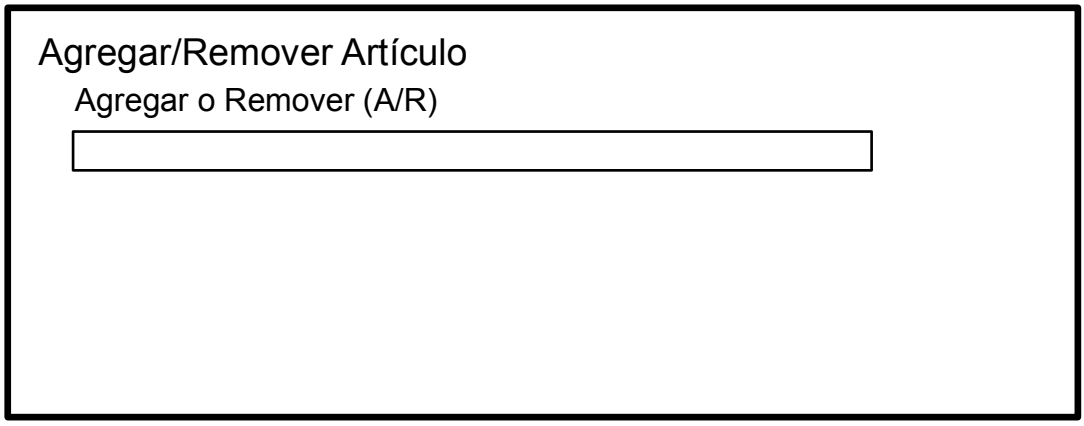
Feedback: *“Esta bien pero creo que nos falto incluir un boton para avisos de productos que ya no tengo en el inventario, osea los que ya se me acabaron.”*

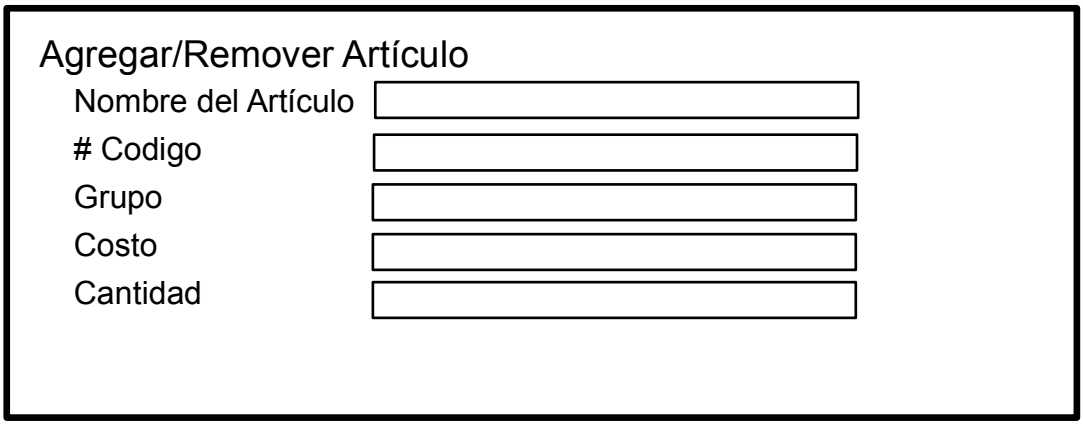
Translation: *“It looks good, but it seems we are missing a button for warnings for any products that I have out of stock, or that I have no more of this certain product.”*

This only acknowledges the user that the list file has been created.

Feedback: *“Ok pero en donde me crea el archivo, es posible crearlo en el desktop para encontrarlo facilmente y luego imprimirlo; tambien si es posbile si puedo escoger el area de los productos que quiero ver la lista y no tener que imprimirla toda.”*

Translation: *“Ok but where is the file created, is it possible to create it in the desktop so it can be easy to find and then print it; also if its possible I would like to choose the area of products that I want to see in the list and not having to print the whole list.”*

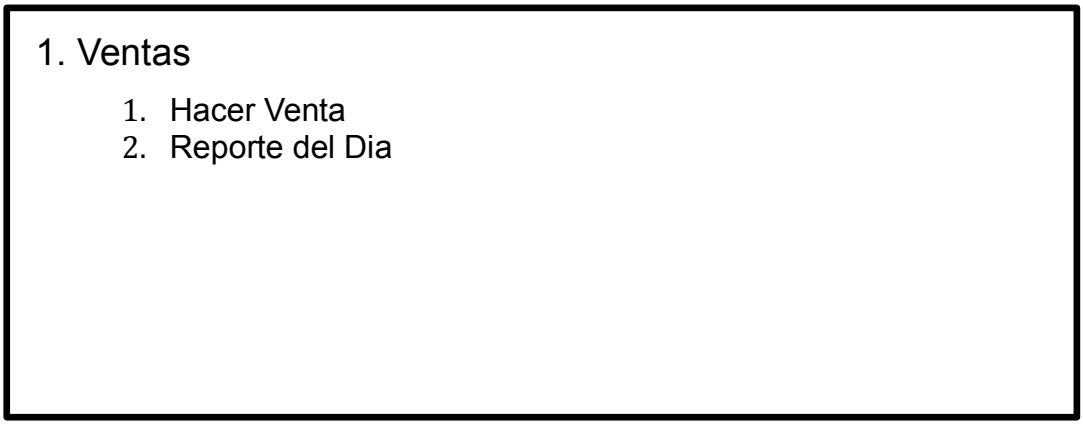




This window demonstrates the adding or removing any product from the inventory, it requires product name, product's code number, group that it belongs to, like screws, pluming, etc., its cost and quantity.

Feedback: *"Muy bien parece que tiene todo lo que necesito ahi solo que cuando quiera remover un articulo no creo que necesite saber el precio o cuantos son solo lo quiero remover de la lista y creo que sería mucho mas facil si solo se escribe el codigo ya que cada product tiene un codigo unico."*

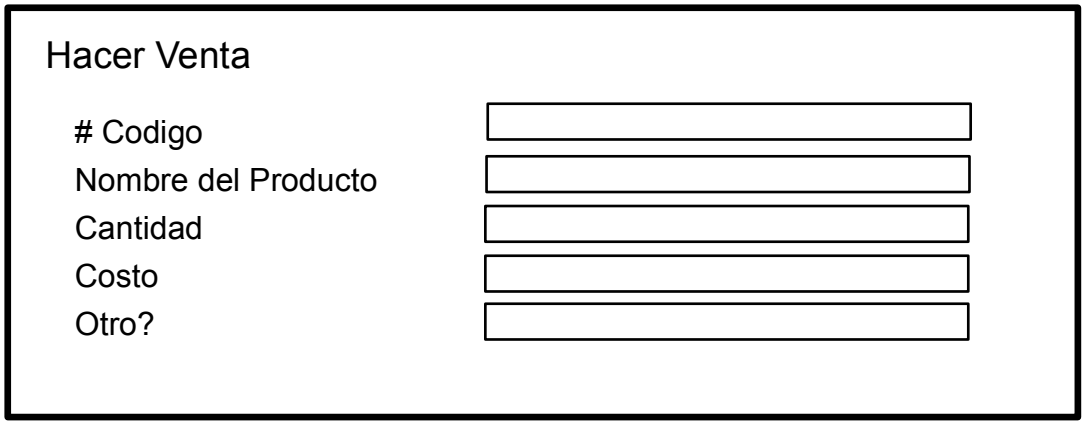
Translation: *"Very good its seems it haves everything I need but when I want to remove a product I do not think that I need to know the price or quantity I just want to remove it from the list and I think that it would be a lot more easier if we only write the code since every product has its unique code."*



This window shows the sales menu.

Feedback: *"Perfecto, me gustaria tener un reporte semanal tambien."*

Translation: *"Perfect, I would also like a weekly report option."*



This window shows the make a sale procedure. First code must be written, then product info appears, if its incorrect, the user writes an ‘n’ do the code can be written again, then it asks quantity and shows cost and the it asks if another product id going to be sold or not, if not total is showed.

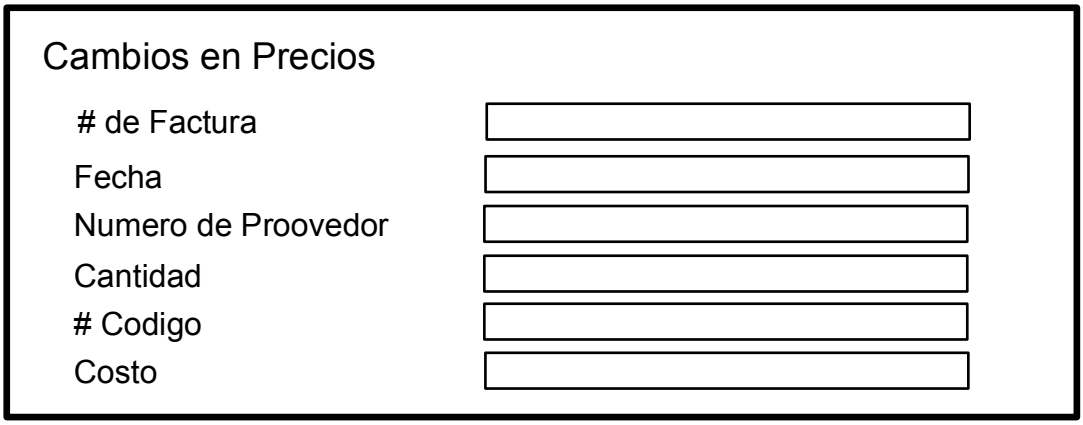
Feedback: *“Bien, me gusto la opcion que me da si quiero cancelar el prodcuto por si me equivoco, muy bien.”*

Translation: *“Good, I liked the option for cancelling a mistake I made, very good.”*

This only acknowledges the user that the list file has been created.

Feedback: *“Bien solo que me gustaria al igual que el otro archivo que se cree el archivo en el desktop para no tener que buscarlo.”*

Translation: *“Good but I would like the file to be created in the desktop as the other file so I can find it easily.”*



This window shows how the price changing procedure works, first invoice number is needed, then date which is automatically written, then supplier number, then quantity, then the code number and finally the cost will be written twice to avoid mistakes

Feedback*:* *“Bueno aqui primero me gustaria que al escribir el numero del provedoor que tengo en la lista me aparezca en pantalla para asegurarme que es tal provedoor, luego al escribir el codigo me gustaria ver la informacion del producto para saber que ese es el producto y luego el costo me gusto la idea de tener que escribirlo dos veces para evitar errores.”*

Translation: *“Well first I would like to write the supplier number that I have in my list and then to see on screen his name to ensure that its him, then when the code is written I would like to see the products info to ensure its that product and finally I liked the idea of writing the cost twice to avoid mistakes.”*

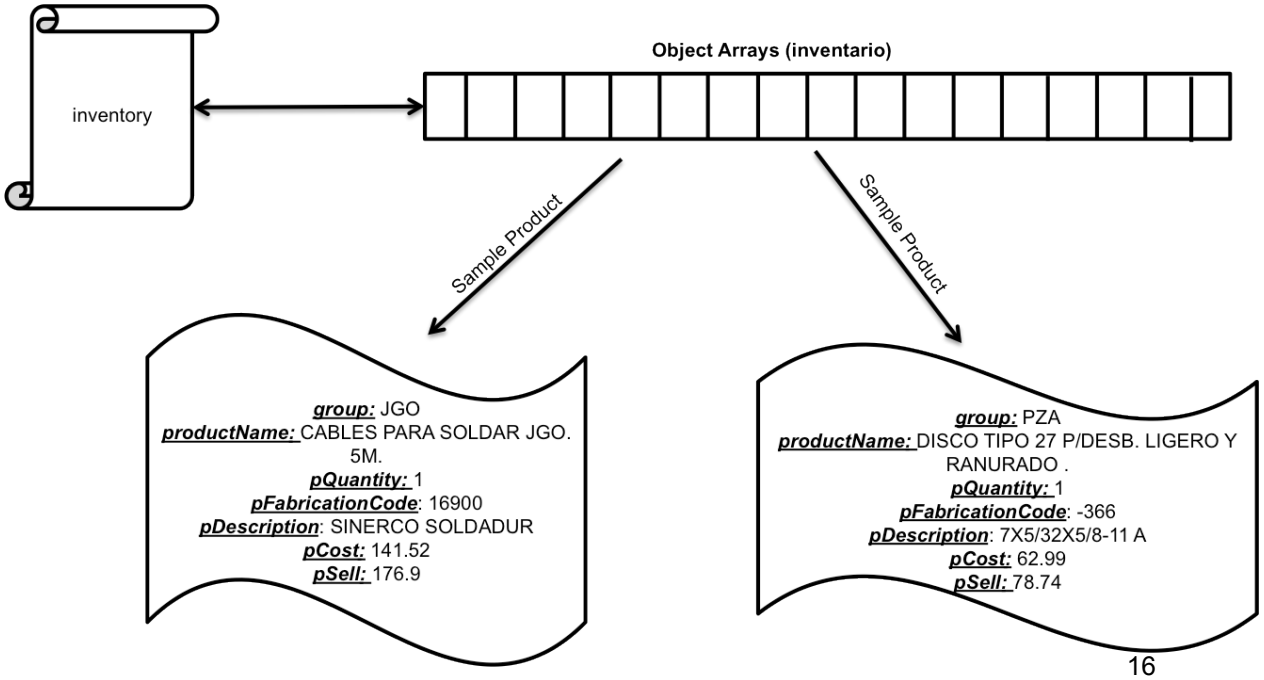
B1. Data Structures

The program my client requires will mainly manage inventory work and management and sales, this will demand many different data structures for it to work in the most proficient way due to its very extensive inventory. An array would be the best option to store the entire inventory since an array can hold many values in a single list so it will make it really easy to manipulate the inventory. This array will need to be an Object array that will hold the diverse data for every product in the inventory. As previously mentioned, every Object will have to hold all of the product's data given that every product has unique and diverse data (different data types) and Object are the only data type that can hold many different data types. The different data types that the array will hold are the following:

Data Type	Name	Discussion	Sample Data
String	group	The group's name that the product belongs to. String was the most suitable option since the name is pure text and numbers and the only data type that can hold a line of text is String.	PZA
String	productName	The product's name. String is the best option since it's the only capable of holding a line of text in one variable.	PEGAMENTO CONTACTO 7000 T-1LT
int	pQuantity	The amount of this product that is in stock. The best data type that could be used in this case is int, since the quantity is only a whole number and there can't be parts of product sold, the best option is int rather than double that handles decimal numbers that will not be used here.	1
String	pFabricationNumber	The number or code for a given product in the inventory. The most suitable option for this will be a String since some of the codes contain letters and numbers, rather than only numbers that could be handled using int.	008FM

String	pDescription	The description of a certain product. This will definitely be a String since a description is a group of words that need to be hold in a variable and String is the only one capable of doing that.	TRUPER
double	pCost	The cost or what my client pays to the supplier for the product. This variable must be a double since the value that is being hold is a money amount and money is used with decimals rater than a whole number so the best option is double.	441.6
double	pSell	The selling price of the product. This variable must be also a double since it is also money that will be handled here and no whole numbers so also the best option is to use a double.	552.0

All of this previously mentioned data must be saved in a File for saving changes in inventory. A Binary File, which will be called completeInventory, would be the most fitting choice for this situation thanks to its ability to easily save a certain data type (array) in it since it transforms data into binary data and not literally as the text file would require only Strings. A Binary File would convert the data more efficiently for the program.

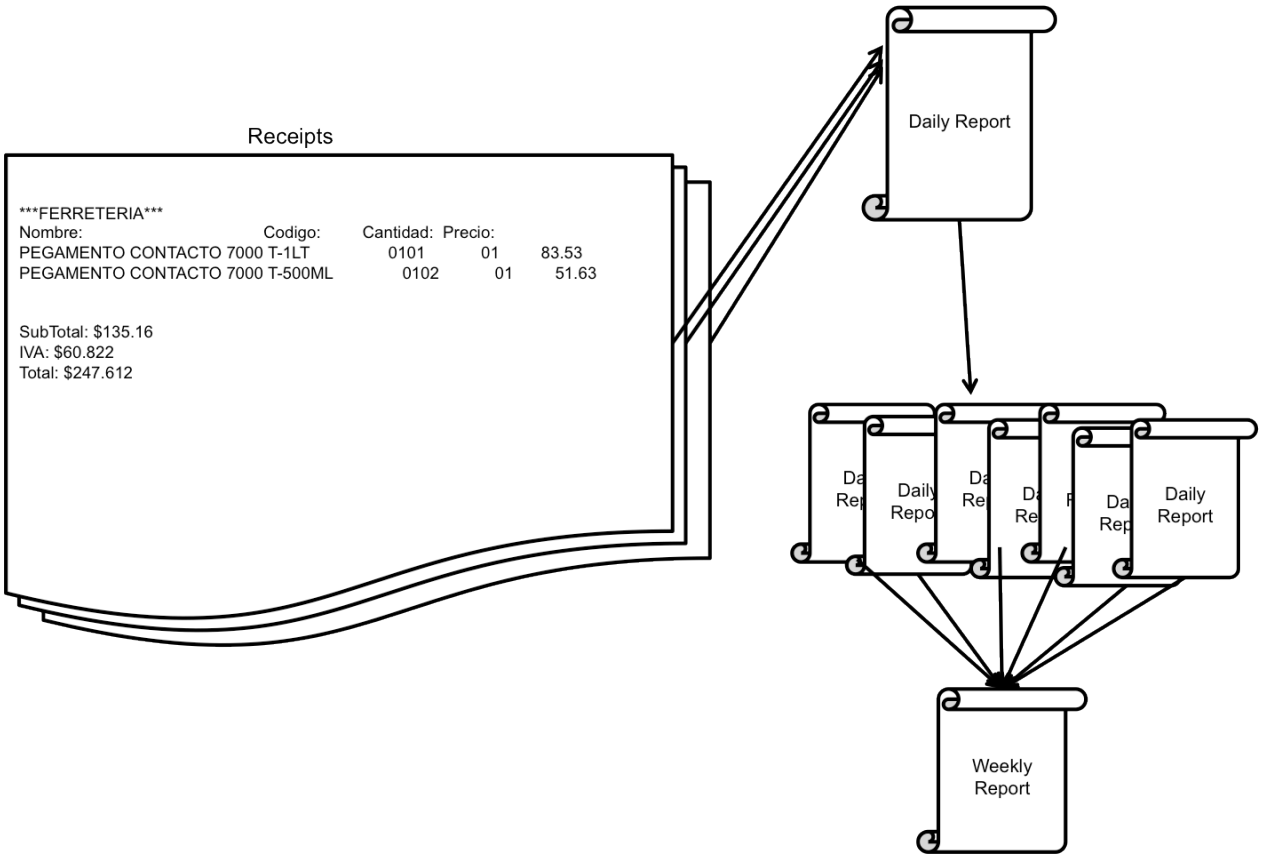


For the sales module the previously mentioned data structures will be used in this module. In this module the user will be capable of making a sale and also create the reports that were requested by my client. Some of the other main data structures that will be used are the following:

Data Type	Name	Discussion	Sample Data
double	subTotal	The total amount before taxes. This variable will need to be a double and not an int because it is handling money number and these numbers need to be handled with decimal in case they are needed no whole numbers.	135.16
double	totalCost	The total amount after taxes. This variable also will need to be a double and not an int since decimals will be used	247.612
double	tax	The total amount that that needs to be paid for taxes. This will also be a double and not an int since decimals are required.	60.822
double	cashFlow	The sum of the total revenue of sales. This needs to be a double since it is the continuous sum of the previous variable totalCost that is a double, so if the previous variable is a double this one must be a double and not an int.	247.612
double	profit	The sum of the difference between price and cost of products. This variable must also be a double since previously price (pSell) and cost (pCost) were stated as doubles. So the data type needs to be the same and an int wont work since it's a different data type and we require decimals.	15.2

int	pSold	The total amount of products sold. This variable will be an int since the value that will be used is a whole number and decimal numbers will not be required so it will no be a double.	4
File	rDiario	Text file that will display the day's revenue, number of products sold and profit. This file needs to be a text file since the user needs to see what is in the file and if the user requires a printed version the file must be text and not binary numbers from a binary file where the user cannot read the file.	
File	rSemana	Text file that will display the week's revenue, number of products sold and profit. As the previous file, this file also need to be a text file so the user can use this file for reading or for printing a copy if needed.	

Next is an illustration of basically how the sales module works. The main process is making a sale, then getting the electronic receipts that will be used to create the daily reports with the stored data in the previously mentioned arrays and finally all of this data will then be used for the weekly reports.

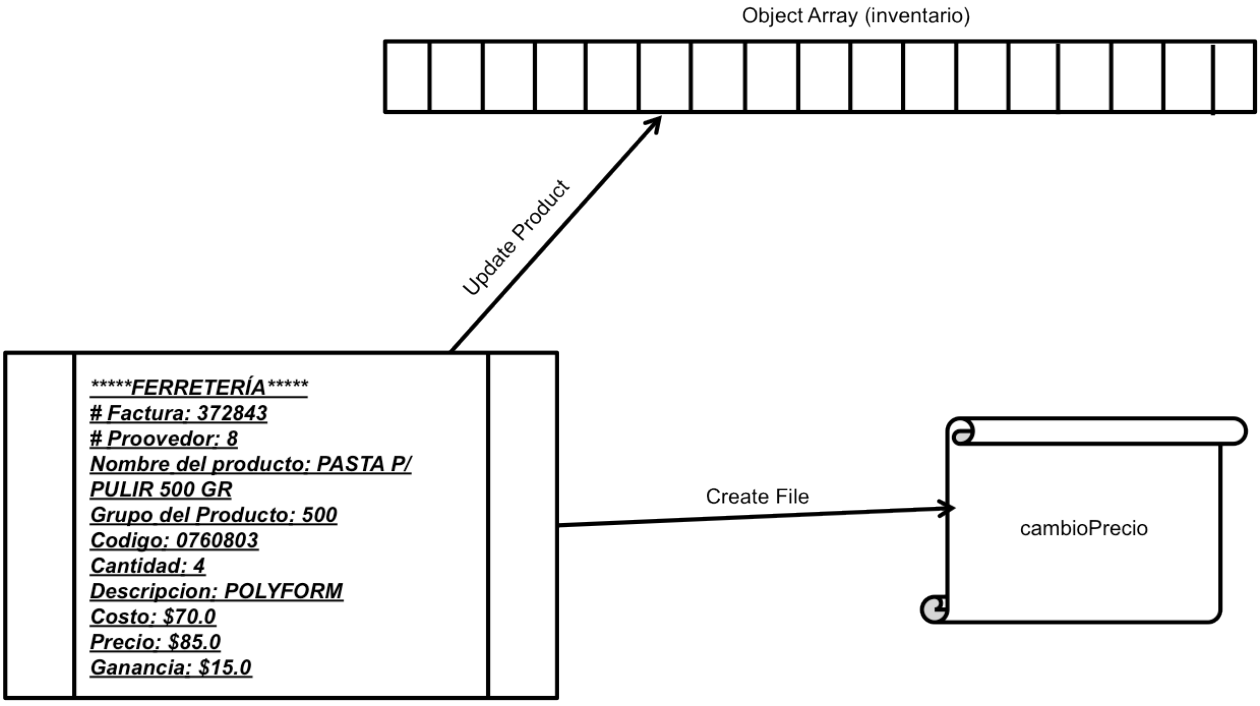


Furthermore the Price Updating division will require some new data structures, which are the following:

Data Type	Name	Discussion	Sample Data
int	invoiceNumber	The number that identifies every invoice. This variable needs to be an int since all of the invoice numbers are only big whole numbers, no decimals. This means that it will not be a double but an int.	0000100230204343 43230
int	supplierNumber	The number that identifies the supplier. This variable will also be an int since the supplier numbers are only whole numbers and no decimals; therefore it cannot be a double.	12
String	date	The date. This variable will be a String since the date is a combination of numbers and	22.11.12

		characters and the String can handle both in one variable.	
File	cambioPrecio	Text file that displays all the information regarding the Price Updating process. This file needs to be a text because the user must be able to read it and a binary file cannot be read so a text file is the best choice.	

Also this is an illustration that basically explains what is being done in this section. First data is gathered for the system then to create the report and updating the inventory.



The following are the previously mentioned data types and other important data types:

Data Type	Name	Brief Description	Sample Data
Application Class	ferreteria.java	Main archive where all the code is saved.	
Object Class	product.java	Archive where code for object is saved.	

Product[]	inventory[]	Array that holds all the product's data	
String	group	The group's name that the product belongs to.	PZA
String	productName	The product's name.	PEGAMENTO CONTACTO 7000 T-1LT
int	pQuantity	The amount of this product that is in stock.	1
String	pFabricationNumber	The number or code for a given product in the inventory.	008FM
String	pDescription	The description of a certain product.	TRUPER
double	pCost	The cost or what my client pays to the supplier for the product.	441.6
double	pSell	The selling price of the product.	552.0
double	subTotal	The total amount before taxes.	135.16
double	totalCost	The total amount after taxes.	247.612
double	tax	The total amount that that needs to be paid for taxes.	60.822
double	cashFlow	The sum of the total revenue of sales.	247.612
double	profit	The sum of the difference between price and cost of products.	15.2
int	pSold	The total amount of products sold.	4
File	rDiario	Text file that will display the day's revenue, number of products sold and profit.	
File	rSemana	Text file that will display the week's revenue, number of products sold and profit.	

int	invoiceNumber	The number that identifies every invoice.	000010023020434343230
int	supplierNumber	The number that identifies the supplier.	12
String	date	The date.	22.11.12
File	cambioPrecio	Text file that displays all the information regarding the Price Updating process.	
boolean	exit	Decision that defines to exit or not the program.	True
int	choice	The number the user chose in a certain menu.	2
char	answer	The letter the user chose to anwer yes or no questions	Y
File	textInventory	Excel file that holds all of the inventory data	
String	menu	The different menus that are displayed for the user	"****VENTAS****"; 1) Hacer Venta
int []	spaces[]	Array that holds amount of spaces for creating a nice and clean text file that contains several columns that need to be organized.	spaces[0] =
int []	days[]	Array that holds day numbers.	days[0] = 22
File	salesData1	Binary file that contains the data for the variable cashFlow.	
File	salesData2	Binary file that contains the data for the variable profit.	
File	salesData3	Binary file that contains the data for the variable pSold.	
File	daysReport	Binary file that contains the data for the array days[].	

int	month	The current month.	11
int	year	The current year.	12
int	day	The current day.	22

B2. Algorithms:

Name: createList	
Description: This method creates a Text file with the entire product's from the groups specified by the user.	
Parameters: spaces[], inventory[]	**Return Value**: No return values

1. Input desired group.
2. Obtain necessary data from inventory[].
3. Create Text File using spaces[] to arrange text.

Name: add	
Description: This method adds a product to the inventory.	
Parameters: inventory[], spaces[]	**Return Value**: inventory[]

1. Input desired group.
2. Input the product's name.
3. Input product's code number.
4. Input product's quantity.
5. Input product's description.
6. Input product's cost.
7. Input product's price.
8. Input product's price again.
9. If prices are the same.
10. If prices are the same.
    - Continue with step 12.
11. Else
    - Repeat step 8.
12. Display gathered information from the user using spaces[] to arrange the text.
13. Answer if displayed data is correct.
14. If data is correct
    - Continue with step 16.

15. Else
    - Repeat step 1.
16. Save new product data in inventory[].
17. Return inventory[].
18.

Name: remove

Description: This method removes a product to the inventory.

Parameters: inventory[], spaces[]	**Return Value**: inventory[]

1. Input product's code.
2. If found
    - Continue with step 4.
3. Else
    - Repeat step 1.
4. Display data using spaces[] to arrange text.
5. Answer if displayed data is correct.
6. If data is correct
    - Define productName as "ZZZZZ"
    - Define pDescription as "ZZZZZ"
    - Define group as "ZZZZZ"
    - Define pFabricationNumber as "ZZZZZ"
    - Define pCost as 0
    - Define pSell as 0
    - Define pQuantity as 0
    - Replace old product's data with previous variables.
7. If data is incorrect
    - Answer if redoing the process would be in order
    - If yes

- Repeat step 1

- If not
  - Continue with step 7.

8. Return inventory[].

Name: notAvailable

Description: This method displays any product that is currently out of stock.

Parameters: inventory[], spaces[]	**Return Value**: No return values

1. Search for products that have quantity in 0.
2. Display found products.

Name: makeSale

Description: This method makes a sale and creates a corresponding receipt.

Parameters: inventory[], spaces[], salesData1, salesData2, salesData3	**Return Value**: No return values

1. Input code number.
2. Search for product.
3. If found
   - Continue with step 5.
4. Else
   - Repeat step 1.
5. Display product's data using spaces[] to organize text.
6. Input Quantity.
7. Modify the product's quantity in inventory.
8. Calculate the subTotal.
9. Calculate the tax.
10. Calculate the totalCost.
11. Create text file with used data (receipt).
12. Add totalCost to cashFlow.

13. Add the difference of pSell and pCost to profit.
14. Add the quantity to pSold.
15. Save cashFlow, profit and pSell in different Binary Files.

Name: reports

Description: This method controls the creation of daily and weekly sales in the business.

Parameters: inventory[], spaces[], daysReport, cashFlow, profit, pSold	**Return Value**: No return values

1. Input type of report D/W (daily or weekly)
2. If input is D
   - Obtain cashFlow, profit and pSold.
   - Create daily report Text File.
3. If input is W
   - Obtain cashFlow, profit and pSold.
   - Create weekly report Text File.

Name: priceUpdating

Description: This method is capable of changing/updating prices when it is needed (when merchandise is received)

Parameters: inventory[]	**Return Value**: No return values

1. Input invoiceNumber.
2. Obtain date.
3. Input supplierNumber.
4. Input product's code number.
5. If found
   - Display product's data.
   - Input new quantity.
   - Input cost.
   - Input price.

- Input price again
- If price is not equal
    - Repeat step (“Input Price”).
- Create cambioPrecio File.

6. Else
    - Repeat step 1.