# Bingo Office Supplies Sales Management Software

Bingo Office Supplies requires software to calculate and record their sales, update stock quantities after sales and output sales and product reports at the close of each business day.

Bingo provided a text file of their current product offering which included product names, current stock availability and cost price.

The cashier software reads this text file at the opening of the program and stores the information in order for sales to be processed. Product names are stored into a listbox on the UI form.

## Calculating the Retail Price

Once a Bingo employee selects a product to be sold from the listbox, a markup of 60% is applied to the product cost price. This is calculated in the Sales Invoice class when the retail price is set. This calculation requires the decimal costPrice from the Product text file. This is calculated in the setSellPrice method as shown: sellPrice = costPrice \* 1.6m. The result is then output to lbl\_prodPrice on the form.

The corresponding picture of the product selected is also output to a picture box on the form.

## Selling the Product

After the retail price and product picture is displayed, a Sell Product button starts the sale process. Once clicked, an invoice number is generated by incrementing the int invNum variable for every button click. This button also reveals the numeric updown form item which allows the user to enter the quantity required for the sale.

## Entering the Quantity

Once the numeric updown value is changed, a Stock Available method is run to ensure that the quantity being sold is available in the current stock of the product. This method checks the int productStockNumber from the Product class against the quantity being entered into the numeric updown (using local variable int quant). An if/else statement is called and if the quantity is available in stock then the sale can proceed. If there is not sufficient stock, a message will display to the user advising the sale cannot proceed. The current stock number will be displayed to the user in this message.

If the quantity is available in stock then the decimal totalSale variable will be calculated. This is done with the following: totalSale = sellPrice \* sellQuantity.

The total is displayed on the form in the lbl\_totalSale label.

## Cancelling a Sale

At this point in the sales process, the sale can be cancelled. If the Cancel Sale button on the Form is clicked, the invoice number is decremented and the total sale field is cleared. All the buttons and numeric updown is also cleared and the form is reset to the beginning of the sales process.

## Confirming a Sale

If the Confirm Sale button is clicked, then the sale is stored in an instance of the SalesInvoice class and added to the SalesReport array. At this point, the software also checks the value of the total sale.

If the value of the sale is less than $20 then the sales details including the int saleInvNumber, string saleProdName, decimal sellPrice, int sellQuantity, decimal totalSale are all stored in a SalesInvoice object and added to the SalesInvoice array.

If the total sale is greater than or equal to $20 then a 5% discount applies to the total value of sale. The discounted total sale will display in the label lbl\_discTotalSale in the form. This discount is calculated with the following: totalSale = sellBSPrice \* sellBSQuantity \* disc. In accordance with Bingo selling conditions, in order for the Big Spender discount (5%) to apply then the customer’s name and phone is required. Once the discounted total price is displayed, then text fields requiring the name and phone number of the customer are revealed in the form. These text boxes require an entry for the sale to be stored. If these fields are empty when the Save button is clicked an error message will appear advising the user that entries are required. Once the Save button is clicked with the filled text boxes, then a BigSpender object is stored with int saleInvNumber, string saleProdName, decimal sellPrice, int sellQuantity, decimal totalSale, string bsCustomerName, string bsCustomerPhone and added to the SalesInvoice array.

If the sale meets the Big Spender conditions and also contains the product of the day, then a further discount will apply. The above Big Spender process will be followed but a further discount will be applied. The SpotPrize discount is a discount of 50% for 1 product of the day. This discount is calculated with the following: \_\_

The discounted total sale including the SpotPrize discount is displayed in the lbl\_discTotalSale. As the SpotPrize discount is only applicable to Big Spender sales then the customer name and phone number is also required.

Once the Save button is clicked, then a SpotPrize object is stored with int saleInvNumber, string saleProdName, decimal sellPrice, int sellQuantity, decimal totalSale, string bsCustomerName, string bsCustomerPhone and added to the SalesInvoice array.

Once the sale is stored then the form clears and the user is able to repeat the sales process.

## Close of Business

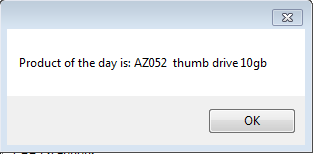
At the close of each business day, a Bingo employee is to click Close of Business. This closes the cashier and reports on the days sales and also updates the stock inventory for each product after the day’s sales. This is to allow the stock to accurately update when the cashier is reopened the following day.

This Close of Business function generates two reports in the form of text files. A Daily Sales report details each of the day’s sales and also resets the invoice number for the following day. This report also total the daily sales for accounting purposes.

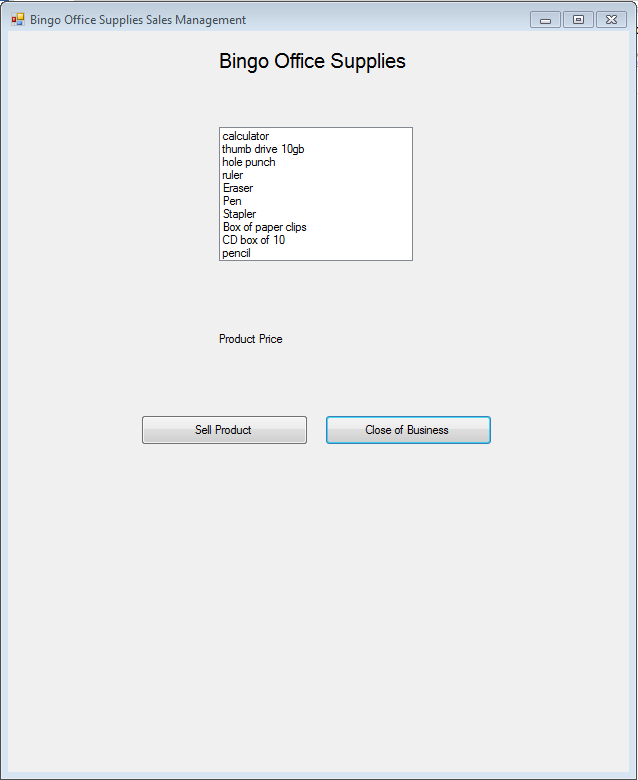
The product text file re-writes the product text file and only updates the stock number taking into account the daily sales.

## Form Layouts

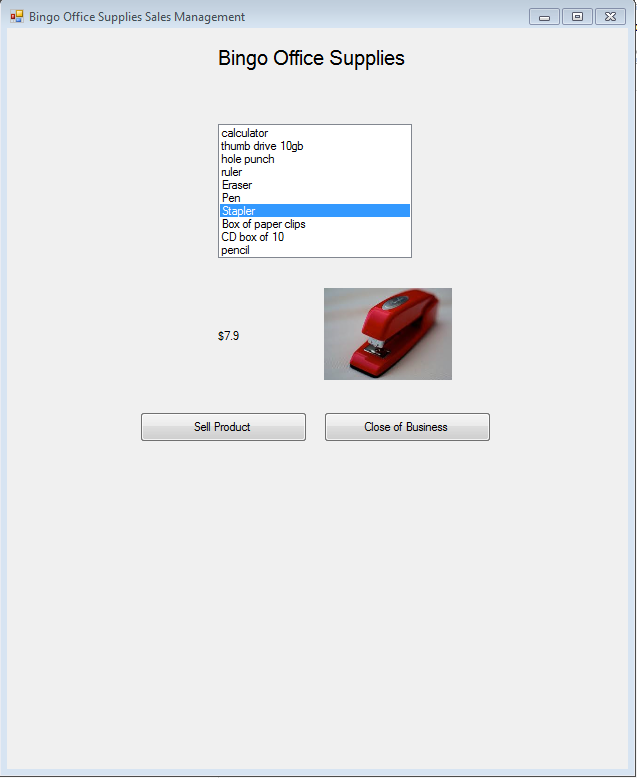
Upon loading the software the Product of the day is shown the Bingo employees



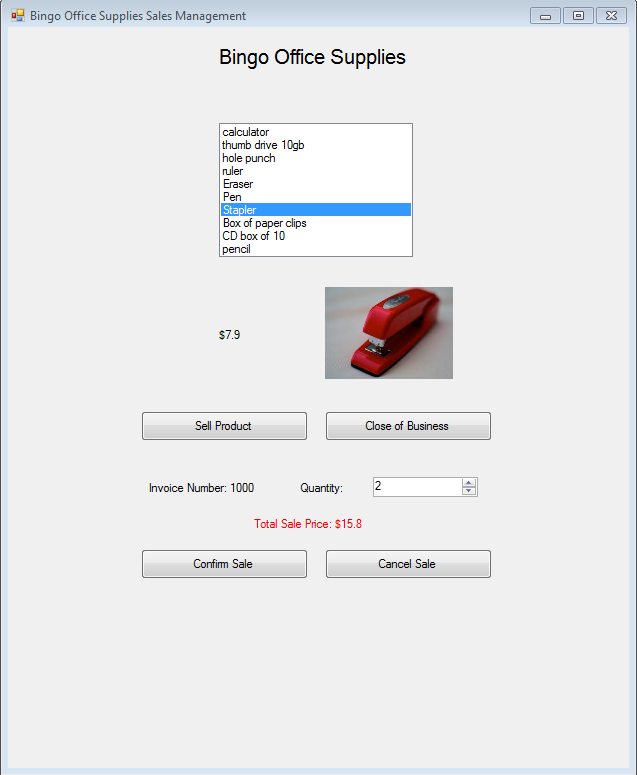
The initial view of the form shows the list of products in the listbox



Once a product is selected then the retail price and the picture is displayed.

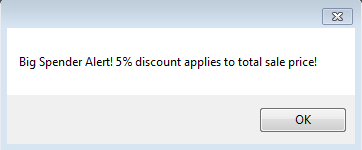


Once the Sell Product button is clicked then the invoice number is displayed. The quantity to be purchased is input via the numeric updown and the total sale price is displayed.

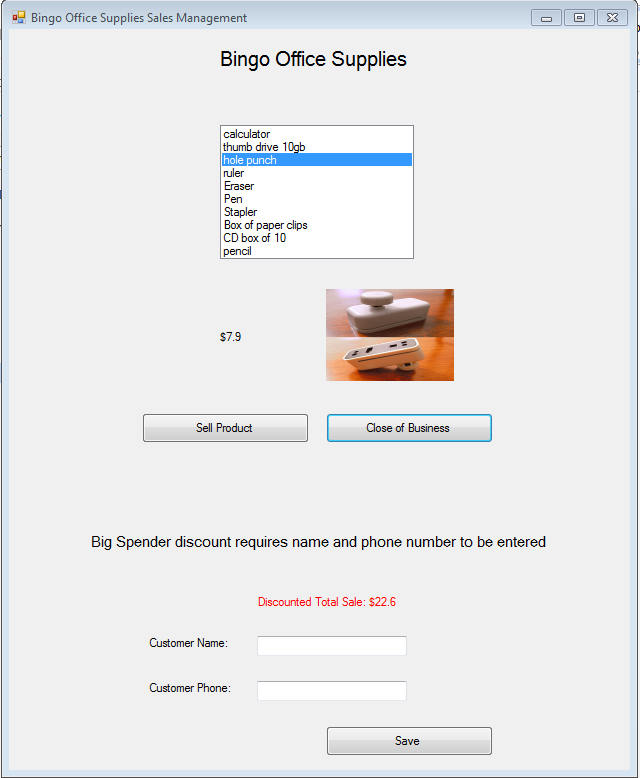


The Confirm Sale will store the sale details. The Cancel Sale button will ignore the sale.

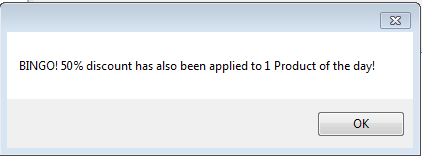
If the sale is equal or greater than $20 then an alert will appear



The discounted total price is displayed and the text fields for the customer name and phone number appear on the form to be filled.

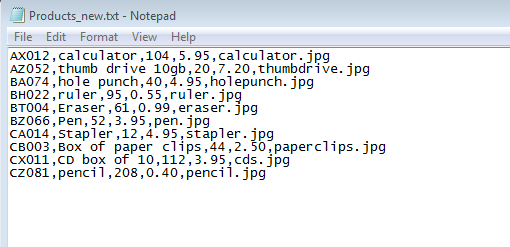


If the product of the day has been sold then an additional alert will appear to the user.

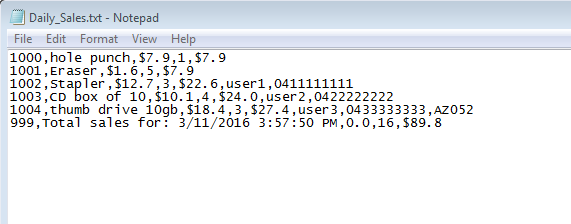


Once the Close of Business button has been clicked then the two text files are generated

The updated Product text file is output as:



The Daily Sales text file is output as:



## Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Input Value** | **Process** | **Expected Output** | **Actual Output** |
| Decimal sellPrice | decimal productCost = 5.95  markup = 1.6 | productCost \* markup | $9.52 | $9.5 |
| Decimal totalSale | Decimal sellPrice = 9.52  Integer sellQuantity = 2 | sellPrice \* sellQuantity | $19.04 | $19 |
| Integer newStock | Integer productStockNumber = 40  Integer sellQuantity = 4 | productStockNumber - sellQuantity | 36 | 36 |
| Decimal spotPrizeTotalSale | Decimal sellBSPrice = 11.52  Integer sellBSQuantity = 3  Decimal disc = 0.95  Decimal spotPrizeDisc = 0.5 | ((sellBSPrice \* 1 \* spotPrizeDisc) + ((sellBSQuantity - 1) \* sellBSPrice)) \* disc | $27.36 | $27.4 |
| Decimal discTotalSale | Decimal sellBSPrice = 9.52  Integer sellBSQuantity = 3  Decimal disc = 0.95 | sellBSPrice \* sellBSQuantity \*  disc | $27.13 | $27.1 |
|  |  |  |  |  |
|  |  |  |  |  |

The actual output of the testing is positive due to the rounding required for a cash only business. If other forms of payment were accepted then non-rounded output would be required.