BUSINESS PROFESSIONALS of AMERICA
Giving Purpose to Potential

C# PROGRAMMING (330)

NATIONAL 2025

PRODUCTION:

National_CSharp ____________(715 points)

Test Time: 90 minutes

Solution and Project (There is NO partial credit) *(NOTE: UC represents uppercase and LC represents lowercase)*		
The VS project file is present on the flash drive in a single folder with your contest ID		10 points
Program Execution (*If the program does not execute, then the remaining items in this section receive a score of zero*)		
Message Box appears showing the file pathway to where the text files are located and displays the number of files in the folder.		30 points
Form displays and the BPA logo is removed from panel.		10 points
STEP 1: Enter LIST command causing list to be generated, vertical scrollbar appears and allows user to view all records.		10 points
Generated list has 9 Dairy and 9 Produce products		50 points
Format of the displayed list matches sample output in terms of the order of printed information: first the statement with number of food-products, statement with estimated value, and the data fields for each record (all must be present and in proper order).		30 points
In the list: **estimated value**, **Total Value, COST of Goods Sold** are in US currency; **Days of Sales Inventory** is an integer (no double values).		20 points
STEP 1: Wrong command is entered in **txtCommand** and message box with statement “Command Entry Error” appears. Closing message box returns cursor focus command text entry box and clears previous entry		20 points
STEP 1: **“Distro”** command (UC or LC) prompts next label to appears asking which food-product to sell and **txtSymbol** text box appears with the cursor focus moves to STEP		10 points
STEP 2: Food-Product symbol (UC or LC) is entered prompts next label to appears asking which food-product to sell and **txtKG_QA** text box appears with the cursor focus moves		10 points
STEP 3: Kilograms are entered for a food-products to sell, and the list changes the information for the food-product dynamically adjusts (estimated value, Total kg, Total Value, Days of Sales Inventory, COST of Goods Sold) moves to STEP 4		10 points
STEP 1: **“Import”** command (UC or LC) prompts next label to appears asking which food-product to sell and **txtSymbol** text box appears with the cursor focus moves to STEP		10 points
STEP 2: Food-Product symbol (UC or LC) is entered prompts next label to appears asking which food-product to mine and **txtKG_QA** text box appears with the cursor focus moves		10 points
STEP 3: Kilograms are entered for a food-product to distro, and the list changes the information for the food-product dynamically adjusts (estimated value, Total kg, Total Value, Days of Sales Inventory, COST of Goods Sold) moves to STEP 4		10 points
STEP 2: Wrong symbol is entered in **txtSymbol** and message box with statement “You have entered incorrect symbol” appears. Closing message box returns cursor focus command text entry box and clears previous entry		20 points
STEP 3: When importing or distributing: characters are entered in **txtKG_QA** a message box with statement “You have entered wrong data type” appears. Closing message box returns cursor focus command text entry box and clears previous entry		30 points
STEP 3: When selling: 0 or more than the number of kilograms are entered in **txtKG_QA** a message box appears indicating wrong amount. Closing message box returns cursor focus command text entry box and clears previous entry		20 points

STEP 3: When importing: 0, or more than 1000 kilograms are entered in **txtKG_QA** a message box appears indicating wrong amount. Closing message box returns cursor focus command text entry box and clears previous entry		20 points
Solution and Project Subtotal		**/330 Points**

Source Code Review (There is NO partial credit) *NOTE: you must place the comment flag in front of the comment in your code in order to get credit. The comment flag will precede the explanation. For example, if the flag is SC1, your comment must read as "//SC1…" in front of the part of the code being reviewed. Code must work to get credit.*		
A comment containing the contestant number is present at the top of the Form1.cs file		10 points
SC1A-C: **Form1.cs** class ***Form1_Load( )*** Place a SC1 comment by these four items: SC1A by the code that demonstrates the activation of the scroll; SC1B code that shows setting the fonts of all the labels to the Calibri font with the size of 12; SC1C how the background image of the panel was removed; SC1D how the panel gets its color from the background color of the windows setting.		20 points
SC2: **Form1.cs** class ***readFile( )*** Place a in code where the file names are read dynamically to only read text files they have the capital word "Text" in them.		40 points
SC3: **Form1.cs** class ***readFile( )*** Place in code where the message box is created and is given a path location to one of the files that will be read.		30 points
SC4A-D: **Form1.cs** class ***readFile( )*** Place a comment by these four sections: **SC4A** code demonstrates reading of the file, recognizing the flags P or D; **SC4B** Code that recognizes the word "stop"; **SC4C** code that handles the construction of the objects; **SC4D** code for separating the fields from reading the file.. No partial credit		40 points
SC5: **Form1.cs** class ***txtCommand_KeyPress ( )*** code showing user input being gathered and passing control to the ***Inventory getCommand( )***		10 points
SC6: **Form1.cs** class ***txtSymbol_KeyPress ( )*** code showing user input being gathered and passing control to the ***Inventory getCommand( )***		10 points
SC7: **Form1.cs** class ***txtTonnage_KeyPress ( )*** code showing user input being gathered and passing control to the ***Inventory getCommand( )***		10 points
SC8: **Inventory.cs** class ***getInventoryTotalValue( )*** code shows how all of the food-products Total Value are sum to show the estimated value of farming operation		20 points
SC9: **Inventory.cs** class ***getInventoryList( )*** code shows how all of the food-products are printed as a list to the ***lblList*** object. Must show how estimated value is formatted into currency.		20 points
SC10: **Inventory.cs** class ***supplySKUVerification( )*** code shows how all of the food-product symbols are checked against user input		20 points
SC11: **Inventory.cs** class ***dataEntryVerification( )*** code shows how all of the food-product ton entries are checked against parameters of selling vs. importing.		20 points
SC12: **Inventory.cs** class ***procureInventory( )*** demonstrates how the data entered into ***txtKG_QA*** is checked for nonnumerical entries. Must use a ***try catch*** exception handler.		20 points
SC13: **Dairy.cs** and **Produce.cs** classes calculate Total Value with ***getValue( )***. Place a SC11 comment by these two sections		10 points
SC14: **Dairy.cs** and **Produce.cs** classes calculate the Days of Sales Inventory (no hard coded data allowed, variables only) with ***DaysSalesOfInventory( )***		10 points

SC15: **Dairy.cs** and **Produce.cs** classes calculate the Cost of Goods Sold (no hard coded data allowed, variables only) with ***costOfGoodsSold( )***.		10 points
SC16: **Dairy.cs** and **Produce.cs** classes ***ToString( )*** method uses string interpolocation to print all 8 of the required data values.		30 points
Source Code Review Subtotal		**/330 Points**
Combined Sub Total Points		**/660 Points**

Calculator Challenge

You will create a basic four-function calculator using your C# in a Windows Form Application with .NET framework project (this will be done your current project). Instructions: Your task is to design and implement a simple five-function (includes modulus) calculator with the following functionalities:

1. User Interface Design:
    - Add a new form to the project, it needs to be called Form2.
    - Design the layout of the calculator interface with buttons and labels.
2. Display Feature:
    - Include a display feature to show the input and output of the calculator operations.
3. Basic Operations:
    - Implement addition, subtraction, multiplication, and division operations.
    - Include buttons for each operation (+, -, *, /, %).
    - Ensure the calculator performs these operations accurately.
4. Numeric Buttons:
    - Include buttons for digits 0-9 to input numbers.
    - Users should be able to input multi-digit numbers.
5. Clear Button:
    - Implement a clear button to reset the calculator display and any ongoing calculation.
6. Equal Button:
    - Include an equal button (=) to perform the calculation and display the result.
7. Error Handling:
    - Implement error handling for invalid input or operations (e.g., division by zero).
    - Program must give the error message “Divide by Zero”
8. Starting the Form2
    - Form2 will start when you press the calculator button that is on the bottom of Form1. This is the only way that Form2 should load. You should be able to close Form2 and return to Form1 at any point in time.

Calculator Challenge		
Form2 loads when the btnCalcuLaunch is pressed		5 points
Form2 UI Calculator has all required buttons		5 points
Form2 when pressing any numeric key the numbers appear in the display		5 points
Form2 Calculator’s adding function works		5 points
Form2 Calculator’s subtracting function works		5 points
Form2 Calculator’s multiplication function works		5 points
Form2 Calculator’s division function works		5 points
Form2 Calculator’s modulus function works		5 points
Form2 Calculator’s equal function works		5 points
Form2 Calculator’s clear function works		5 points
Form2 when attempting to divide by zero the calculator will not allow action. You must provide a user with an “Divide by Zero” message		5 points
Calculator Challenge		**/55 Points**
Solution and Project		**/330 Points**
Source Code Review		**/330 Points**
Grand Total		**/715 TOTAL**

Solution Code

```
... National\C# National Grader\National_CSharp\Form1.cs                    1
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace RestaurantSupplier_National
{
    public partial class Form1 : Form
    {

        String foodCategory;
        String FoodSymbol;
        String FoodName;
        double FoodCost;
        int FoodVolume;
        Inventory foodStock;



        public Form1()
        {
            InitializeComponent();

        }

        private void Form1_Load(object sender, EventArgs e)
        {

            panel1.AutoScroll = true;  //SC1A
            panel1.BackColor = SystemColors.Window; //SC1D
            lblEstimatedValue.Font = new Font("Calibri", 12, FontStyle.Bold | FontStyle.Italic); //SC1B
            lblTotalCounts.Font = new Font("Calibri", 12, FontStyle.Bold | FontStyle.Italic);//SC1B
            lblList.Font = new Font("Calibri", 10, FontStyle.Bold); //SC1B
            panel1.BackgroundImage = null; //SC1C



            foodStock = new Inventory(this);
            lblGreeting.Visible = true;
            txtCommand.Visible = true;
            lblGreeting.Text = "What do you want to do today: IMPORT,
              DISTRO, LIST, or END ? ";
            txtCommand.Focus();
            readFile();

        }


        private void readFile()
        {

            string resourceFolderPath = Path.Combine(Path.GetDirectoryName
              (System.AppDomain.CurrentDomain.BaseDirectory),
              "Resources");

            string[] files = Directory.GetFiles(resourceFolderPath,
              "*Text*.txt"); //SC2
            MessageBox.Show("FILEPATHWAY HELP MESSAGE \nYour file location
               and count: " + files[0],  "Total readable
              files:"+files.Length); //SC3
            foreach (string file in files)
            {
                String temp = "";
                int counter = 0;
                String temp2 = "";
                String temp3 = "";

                foreach (String line in System.IO.File.ReadLines(file))
                {
                    temp += line;
                }
                foreach (Char c in temp)
                {
                    temp2 += c;
                    if (temp2.ToUpper().Equals("STOP") == false)  //SC4B
                    {
                        if (temp2.Contains(",") == true)  //SC4D
                        {
                            if (counter % 5 == 0)
                            {
                                temp3 = temp2.Remove(temp2.Length - 1,
                   1); // Check to see if it removes the comma from the end
                    of the string
                                foodCategory = temp3;
                                temp2 = "";
                                temp3 = "";
                                counter++;
                            }
                            else if (counter % 5 == 1)
                            {
                                temp3 = temp2.Remove(temp2.Length - 1, 1);
                                FoodSymbol = temp3;
                                temp2 = "";
                                temp3 = "";
                                counter++;
                            }
                            else if (counter % 5 == 2)
                            {
                                temp3 = temp2.Remove(temp2.Length - 1, 1);
                                FoodName = temp3;
                                temp2 = "";
                                temp3 = "";
                                counter++;
                            }
                            else if (counter % 5 == 3)
                            {
                                temp3 = temp2.Remove(temp2.Length - 1, 1);
                                Double.TryParse(temp3, out FoodCost);
                                temp2 = "";
                                temp3 = "";
                                counter++;
                            }
                            else if (counter % 5 == 4)
                            {
                                temp3 = temp2.Remove(temp2.Length - 1, 1);
                                int.TryParse(temp3, out FoodVolume);
                                temp2 = "";
                                temp3 = "";
                                counter++;
                                if (foodCategory.ToUpper().Equals("P")) // SC1A
                                {
                                    foodStock.produceListAdditions(FoodSymbol, FoodName, FoodCost, FoodVolume); //SC4A
                                }
                                else if (foodCategory.ToUpper().Equals("D")) //SC1A
                                {
                                    foodStock.dairyListAdditions(FoodSymbol, FoodName, FoodCost, FoodVolume); //SC4A
                                }
                            }
                        }
                    }
                }
            }
        }

        //Gathers information from the text box when enter key is pressed.
```

```
... National\C# National Grader\National_CSharp\Form1.cs                    4
        private void txtCommand_KeyPress(object sender, KeyPressEventArgs e)  //SC5
        {
            if (e.KeyChar == Convert.ToInt16(Keys.Enter))
            {
                foodStock.getCommand();
                e.Handled = true; //Stops dinging Windows sound

            }

        }

        //Gathers information from the text box when enter key is pressed.
        public void txtSymbol_KeyPress(object sender, KeyPressEventArgs e)  //SC6
        {
            if (e.KeyChar == Convert.ToInt16(Keys.Enter))
            {
                if(txtCommand.Text.ToUpper().Contains("DISTRO"))
                {

                    foodStock.distro_InventoryDataEntry();
                    e.Handled = true;
                }
                else if(txtCommand.Text.ToUpper().Contains("IMPORT"))
                {

                    foodStock.procure_InventoryDataEntry();
                    e.Handled = true;
                }
            }

        }

        //Gathers information from the text box when enter key is pressed.
        private void txtTonnage_KeyPress(object sender, KeyPressEventArgs e)  //SC7
        {
            if (e.KeyChar == Convert.ToInt16(Keys.Enter))
            {
                if (txtCommand.Text.ToUpper().Contains("DISTRO"))
                {
                    foodStock.distro_InventoryFiles();
                    e.Handled = true;
                }
                else if (txtCommand.Text.ToUpper().Contains("IMPORT"))
                {
                    foodStock.procureInventory();
                    e.Handled = true;
... National\C# National Grader\National_CSharp\Form1.cs                       5
                }
            }

        }

        private void txtSymbol_TextChanged(object sender, EventArgs e)
        {

        }
    }
}

```

```
using System;
using System.Diagnostics;
using System.Globalization;
using System.Xml.Linq;

namespace RestaurantSupplier_National
{

    internal class Dairy: Food
    {
        private String symbol;
        private String name;
        private double price;
        private int kg;
        private String type = "Dairy";
        private double cost;

        public Dairy() : base()
        {

        }

        public Dairy(String sym, String nam, double pri, int kg)
        {
            symbol = sym;
            name = nam;
            price = pri;
            this.kg = kg;
        }


        public override double getValue() //SC13
        {
            return price * kg;
        }


        public override double getProfit()
        {
            return ((getValue()-5000)/5000);
        }
        //SC14
        public override double DaysSalesOfInventory()
        {

            return (int)((kg / costOfGoodsSold()) * 365); // Assuming 
              annual sales period
        }
        //SC15
        public override double costOfGoodsSold()
        {
            double tenPercent = kg * 0.1;
            Random rnd = new Random();
            cost = (rnd.NextDouble() * (tenPercent - 1) + 1) * 20;

            return cost;
        }
         //SC16
        public override String ToString()
        {
            return $"Food Details:\n" +
           $"- Symbol: {getSKU()}\n" +
           $"- Food Type (P or D): {type}\n" +
           $"- Food Name: {getName()}\n" +
           $"- Price: {getPrice().ToString("C",
             CultureInfo.CurrentCulture)}\n" +
           $"- Total Tons: {kg}\n" +
           $"- Total Value: {getValue().ToString("C",
             CultureInfo.CurrentCulture)}\n" +
           $"- Days of Sales Inventory: {DaysSalesOfInventory().ToString
             ()}" + "\n" +
           $"- COST of Goods Sold: {costOfGoodsSold().ToString("C",
             CultureInfo.CurrentCulture)}" + "\n";

        }

        //Returns number kg
        public override int getTons()
        {
            return kg;
        }

        //Returns symbol
        public override String getSKU()
        {
            return symbol;
        }

        //Returns name
        public override String getName()
        {
            return name;
        }

        //Returns price
        public override double getPrice()
        {
            return price;
        }

        //Adds the number of requested kg
        public override void procureKG(int s)
        {
            kg += s;
        }

        //Sells the number of requested kg
        public override void distroKG(int s)
        {
            kg -= s;
        }


    }
}
```

...# National\C# National Grader\National_CSharp\Food.cs 1

```
using System;

namespace RestaurantSupplier_National
{
    //This entire parent class is complete. You will need to copy the 
      methods and variables to be included in the children classes
    //which is the precious Producec and Dairy.
    internal class Food
    {
        private String symbol;
        private String name;
        private double price;
        private int kg;

        public Food()
        {
            symbol = "###";
            name = "Generic";
            price = 0;
            kg = 0;
        }

        public Food(String s, String n, double p)
        {
            symbol = s;
            name = n;
            price = p;
        }

        public virtual String ToString()
        {
            return "";
        }

        //Returns symbol
        public virtual String getSKU()
        {
            return symbol;
        }

        //Returns name
        public virtual String getName()
        {
            return name;
        }

        //Returns price
        public virtual double getPrice()
        {
                return price;
            }

            //Returns the total $ value of the investment product (no formula)
            public virtual double getValue()
            {
                return 0.0;
            }

            //Returns the return on investment (%) of the investment product 
              (no formula)
            public virtual double getProfit()
            {
                return 0.0;
            }
            //Returns kg
            public virtual int getTons()
            {
                return kg;
            }

            //Adds the number of requested kg
            public virtual void procureKG(int s)
            {
                kg += s;
            }

            //Sells the number of requested kg
            public virtual void distroKG(int s)
            {
                kg -= s;
            }

            public virtual double DaysSalesOfInventory()
            {
                return (kg / costOfGoodsSold()) * 365; // Assuming annual sales 
                   period
            }

            public virtual double costOfGoodsSold()
            {
                double tenPercent = kg * 0.1;
                Random rnd = new Random();
                double cost = rnd.NextDouble() * (tenPercent - 1) + 1;

                return cost;
            }
        }
    }
```

...ional\C# National Grader\National_CSharp\Inventory.cs 1

```
using System;
using System.Globalization;
using System.Collections.Generic;
using System.Data.Common;
using System.Net.NetworkInformation;
using System.Reflection.Emit;
using System.Security.Cryptography;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace RestaurantSupplier_National
{
    internal class Inventory: Form1
    {

        public List<Food> list_Dairy = new List<Food>();
        public List<Food> list_Produce = new List<Food>();
        private int dairyCounter = 0;
        private int produceCounter = 0;
        private Form1 mainform;
        String answerSymbol;
        int answerCount;

        public Inventory(Form1 form1)
        {
            mainform = form1;
        }


        //Creates Produce objects that are added to general list. Counts 
          how many are created.
        public void produceListAdditions(String sku, String nam, double 
          cost, int kg)
        {
            list_Produce.Add(new Produce(sku, nam, cost, kg));
            produceCounter++;

        }


        //Creates Dairy objects that are added to general list. Counts how 
           many are created.
        public void dairyListAdditions(String sku, String nam, double 
          cost, int kg)
        {
            list_Dairy.Add(new Dairy(sku, nam, cost, kg));
            dairyCounter++;
        }

```

```csharp
        //Use this method to get the typed commands from the user and then perform actions based upon the input.
        //This could be a good place to also control how the text appears on the labels.
        public void getCommand()  //SC5
        {
            String commandAnswer;
            mainform.lblTotalCounts.Visible = false;
            mainform.lblEstimatedValue.Visible = false;
            mainform.lblRetypeCommand.Visible = false;

            String[] commands = { "IMPORT", "DISTRO", "LIST", "END" };

            //Tests that the user entry matches the available commands
            mainform.lblGreeting.Visible = true;
            mainform.txtCommand.Visible = true;
            mainform.txtCommand.Focus();
            mainform.lblGreeting.Text = "What do you want to do today: IMPORT, DISTRO, LIST, or END ? ";

            //Add a check method to see whether the enter has been clicked
            commandAnswer = mainform.txtCommand.Text;
            commandAnswer = commandAnswer.ToUpper();

            if (commandAnswer.CompareTo("IMPORT") == 0)
            {
                mainform.txtSymbol.Visible = true;
                mainform.lblFoodCode.Visible = true;
                mainform.lblFoodCode.Text = "Which food-product do you want to procure?";
                mainform.txtSymbol.Focus();

            }
            else if (commandAnswer.CompareTo("DISTRO") == 0)
            {
                mainform.txtSymbol.Visible = true;
                mainform.lblFoodCode.Visible = true;
                mainform.lblFoodCode.Text = "Which food-product do you want to sell?";
                mainform.txtSymbol.Focus();
            }
            else if (commandAnswer.CompareTo("LIST") == 0)
            {
                getInventoryList();
            }
            else if (commandAnswer.CompareTo("END") == 0)
            {
                MessageBox.Show("Goodbye");
                mainform.Close();
            }
            else
            {
                MessageBox.Show("Command Entry Error");
                mainform.txtCommand.Text = "";
            }

        }

        //This will allow the user to type the commands to perform one of 
          the actions
        //The actions are already programmed in advance
        //**list
        private double getInventoryTotalValue()  //SC8
        {
            double tempValue = 0;
            foreach (Food s in list_Dairy)
            {
                tempValue += s.getValue();

            }
            foreach (Food s in list_Produce)
            {
                tempValue += s.getValue();

            }
            return tempValue;
        }
        //**list
        //This is not a required method. It will get all of the print 
          information from the food-product objects.
        public override String ToString()
        {

            String inventoryPrinter = "";
            inventoryPrinter = inventoryPrinter + "\n
              ===========PRODUCE==================\n\n";
            for (int a = 0; a < list_Produce.Count; a++)
            {
                inventoryPrinter = inventoryPrinter + list_Produce
                  [a].ToString() + "\n";

            }
            inventoryPrinter = inventoryPrinter + "\n
              ===========DAIRY==================\n\n";
            for (int a = 0; a < list_Dairy.Count; a++)
            {
                inventoryPrinter = inventoryPrinter + list_Dairy
                  [a].ToString() + "\n";

                }

                return inventoryPrinter;

            }


            //NOTE: all of the label and text box visibility code is optional 
              since this is a very long test.
            //This method prints list
            private void getInventoryList()  //SC9
            {
                mainform.lblTotalCounts.Visible = true;
                mainform.lblList.Visible = true;
                mainform.lblEstimatedValue.Visible = true;
                mainform.lblTotalCounts.Text = "There are " + produceCounter + 
                   " produce food-products and " + dairyCounter + " dairy 
                  food-products.";

                mainform.lblList.Text = ToString();
                mainform.lblEstimatedValue.Text = "This is the estimated value 
                   of the invetory for the restaraunt supplier: " + 
                  getInventoryTotalValue().ToString("C", 
                  CultureInfo.CurrentCulture); //SC9

                mainform.lblFoodCode.Visible = false;
                mainform.lblKG_QA.Visible = false;
                mainform.txtSymbol.Visible = false;
                mainform.txtSymbol.Text = "";
                mainform.txtKG_QA.Text = "";
                mainform.txtKG_QA.Visible = false;


                mainform.lblRetypeCommand.Visible = true;
                mainform.lblRetypeCommand.Text = "Please enter in your next 
                  command.";
                mainform.txtCommand.Text = "";
            }


            //Non required helper method. This is used to control the farming 
              process
            public void procure_InventoryDataEntry()
            {
                MessageBox.Show("procure method runs");
                Boolean verify;
                answerSymbol = mainform.txtSymbol.Text;
                answerSymbol = answerSymbol.ToUpper();
                verify = supplySKUVerification(answerSymbol);
                if (verify==false)
                {
                    MessageBox.Show("You have entered an incorrect symbol");
                    mainform.lblFoodCode.Text = "Incorrect symbol entry. 
                      Reenter the correct symbol.";
                    mainform.txtSymbol.Text = "";
                    mainform.txtSymbol.Focus();
                }
                else
                {
                    mainform.lblKG_QA.Visible = true;
                    mainform.txtKG_QA.Visible = true;
                    mainform.txtKG_QA.Focus();
                    mainform.lblKG_QA.Text = "How many kg do you want to 
                      procure? (NOTE: 1000 kg is the maximum limit)";
                }
            }


            public void procureInventory()
            {
                String answer;
                Boolean verify = false;
                mainform.txtKG_QA.Focus();
                try   //SC12
                {
                    answer = mainform.txtKG_QA.Text;
                    answerCount = Convert.ToInt32(answer);
                }
                catch(FormatException)
                {
                    MessageBox.Show("You have entered the wrong data type");
                    mainform.txtKG_QA.Clear();
                    mainform.txtKG_QA.Focus();
                    return; //escape from the method

                }

                verify = dataEntryVerification(answerCount, answerSymbol, 
                  true); // Check if the variables are reaching this point

                if (!verify)
                {
                    MessageBox.Show("Incorrect Data Entry. Reenter a value 
                      greater than 0 and not greater than 1000");

                    mainform.txtKG_QA.Clear();
                    mainform.txtKG_QA.Focus();
            }
            else if (verify==true)
            {
                exchange(answerSymbol, answerCount, true);
            }
            else
            {
                mainform.lblKG_QA.Text = "Warning Logic Error: This is the final else statement"; //Code should not reach this point
            }
        }

        //Non required helper method. This is used to control the selling process
        public void distro_InventoryDataEntry()
        {
            Boolean verify;
            answerSymbol = mainform.txtSymbol.Text;
            answerSymbol = answerSymbol.ToUpper();
            verify = supplySKUVerification(answerSymbol);
            if (verify == false)
            {
                MessageBox.Show("You have entered an incorrect symbol");

                mainform.txtSymbol.Text = "";
                mainform.txtSymbol.Focus();
            }
            else
            {
                mainform.lblKG_QA.Visible = true;
                mainform.txtKG_QA.Visible = true;
                mainform.txtKG_QA.Focus();
                mainform.lblKG_QA.Text = "How many kg do you want to sell?";
            }
        }

        //called from Form1, used to check data entry. Not required
        public void distro_InventoryFiles()
        {
            String answer;
            Boolean verify = false;
            mainform.txtKG_QA.Focus();
            try   //SC10
            {
                answer = mainform.txtKG_QA.Text;
                answerCount = Convert.ToInt32(answer);
            }
            catch (FormatException)
            {
                MessageBox.Show("You have entered the wrong data type");
                mainform.txtKG_QA.Clear();
                mainform.txtKG_QA.Focus();
                return; //escape from the method and forces user to have to enter in the correct tonnage values
            }

            mainform.label1.Text = answerCount.ToString();
            verify = dataEntryVerification(answerCount, answerSymbol, false); // Check if the variables are reaching this point
            if (!verify)
            {
                MessageBox.Show("Incorrect Data Entry. Reenter a value greater than 0, and less-or-equal to amount in the foodStock");
                mainform.txtKG_QA.Clear();
                mainform.txtKG_QA.Focus();
            }
            else if (verify == true)
            {
                exchange(answerSymbol, answerCount, false);
            }
            else
            {
                mainform.lblKG_QA.Text = "You have entered in the wrong value type";
            }
        }

        //Use this method to verifiy if the user is correctly entering symbols
        private Boolean supplySKUVerification(String sym)  //SC10
        {

            for (int i = 0; i < list_Dairy.Count; i++)
            {
                if ((sym.Equals(list_Dairy[i].getSKU())))
                    return true;
            }
            for (int i = 0; i < list_Produce.Count; i++)
            {
                if ((sym.Equals(list_Produce[i].getSKU())))
                    return true;
            }
            return false;
        }

        //Use this method to verifiy if the user is correctly entering numerical values
        private Boolean dataEntryVerification(int amt, String sym, Boolean procureOR_Distro)  //SC11
        {

            int temp = 0;
            if (procureOR_Distro) //true mines
            {
                if (amt <= 0 || amt > 1000)
                    return false;
                if (procureOR_Distro)
                    return true;
            }

            bool tempHold_TF = false;

            for (int i = 0; i < list_Dairy.Count; i++) //generating a list of
            {
                if ((sym.CompareTo(list_Dairy[i].getSKU()) == 0))
                {
                    temp = list_Dairy[i].getTons();
                    if (amt > temp)
                        tempHold_TF = false;
                    else
                        tempHold_TF = true;
                }

            }
            for (int i = 0; i < list_Produce.Count; i++) //generating a list of
            {
                if ((sym.CompareTo(list_Produce[i].getSKU()) == 0))
                {
                    temp = list_Produce[i].getTons();
                    if (amt > temp)
                        tempHold_TF = false;
                    else
                        tempHold_TF = true;
                }

            }
            return tempHold_TF;
        }

        //Not required method. This determines if the user is importing or distributing, and then adds or subtracts via object methods.
        private void exchange(String sym, int amt, Boolean
            procureOR_Distro) //true mines...false sells
        {

            for (int i = 0; i < list_Dairy.Count; i++) //generating a list of
            {
                if (sym.CompareTo(list_Dairy[i].getSKU()) == 0)
                {
                    if (procureOR_Distro) list_Dairy[i].procureKG(amt);
                    else { list_Dairy[i].distroKG(amt); }
                }
            }
            for (int i = 0; i < list_Produce.Count; i++) //generating a list of
            {
                if (sym.CompareTo(list_Produce[i].getSKU()) == 0)
                {
                    if (procureOR_Distro) list_Produce[i].procureKG(amt);
                    else { list_Produce[i].distroKG(amt); }
                }
            }
            getInventoryList();
        }

    }
}
```

...# National\C# National Grader\National_CSharp\Food.cs 1

```
using System;

namespace RestaurantSupplier_National
{
    //This entire parent class is complete. You will need to copy the methods and variables to be included in the children classes
    //which is the precious Producec and Dairy.
    internal class Food
    {
        private String symbol;
        private String name;
        private double price;
        private int kg;

        public Food()
        {
            symbol = "###";
            name = "Generic";
            price = 0;
            kg = 0;
        }

        public Food(String s, String n, double p)
        {
            symbol = s;
            name = n;
            price = p;
        }

        public virtual String ToString()
        {
            return "";
        }

        //Returns symbol
        public virtual String getSKU()
        {
            return symbol;
        }

        //Returns name
        public virtual String getName()
        {
            return name;
        }

        //Returns price
        public virtual double getPrice()
        {
            return price;
        }

        //Returns the total $ value of the investment product (no formula)
        public virtual double getValue()
        {
            return 0.0;
        }

        //Returns the return on investment (%) of the investment product (no formula)
        public virtual double getProfit()
        {
            return 0.0;
        }
        //Returns kg
        public virtual int getTons()
        {
            return kg;
        }

        //Adds the number of requested kg
        public virtual void procureKG(int s)
        {
            kg += s;
        }

        //Sells the number of requested kg
        public virtual void distroKG(int s)
        {
            kg -= s;
        }

        public virtual double DaysSalesOfInventory()
        {
            return (kg / costOfGoodsSold()) * 365; // Assuming annual sales period
        }

        public virtual double costOfGoodsSold()
        {
            double tenPercent = kg * 0.1;
            Random rnd = new Random();
            double cost = rnd.NextDouble() * (tenPercent - 1) + 1;

            return cost;
        }
    }
}
```