Programmed by: Christopher Singleton   Coke Machine that allows up to five cokes using C#.

Back to Resume

(Classes, Objects, Methods, Fields and Constructors)

Console.cs

```csharp
using ClassLibrary1;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace VendCoke
{
    class Program
    {
        static void Main(string[] args)
        {
            string customerCommand;  // used to hold customer's command
            int customerAmt = 1;     // used to hold deposit or withdrawal amount
            int cokeAmt = 5;        // used to hold coke amount.

            Boolean fundsAvailable; // used to catch result of funds availability.
            Boolean cokeAvailable;  // used to catch result of coke availability.
            ChangeVend BellevueVendCoke;  // name and create a new variable of type ChangeVend, then.
            BellevueVendCoke = new ChangeVend(); //instantiate an object instance of a new object, call
                                                    constructor, new returns ref to object.
            Console.Write("\n\t    Welcome to the Bellevue Coke Machine! \n"); //Give a welcome to the customer.
            BellevueVendCoke.DisplayMenu(); // Display the menu.

            customerCommand = Console.ReadLine();           // Allows user input.
            customerCommand = customerCommand.ToUpper();    // Allow user to type upper or lower case.
            while (customerCommand != "Q")                  // Loop until user is done.
            {
                switch (customerCommand)
                {
                    case "P": // Deposit a dollar.
                        Console.Clear();
                        BellevueVendCoke.AcceptCash(customerAmt); // Add funds using the AcceptCash method.
                        Console.Write("Thank you! You now have ${0}.00 \n", BellevueVendCoke.Balance); //Call the
                                                                                          balance method.
                        BellevueVendCoke.DisplayMenu();
                        break;

                     case "B": // Buy a coke.
                         /* Check true or false if funds and/or coke is available by calling both methods. */
                         fundsAvailable = BellevueVendCoke.CheckPurchase(customerAmt);
                         cokeAvailable = BellevueVendCoke.BuyCoke(cokeAmt);
                         if (fundsAvailable == true && cokeAvailable == true)
                         {
                             Console.Clear();
                             Console.Write("Thank you for your purchase! You have ${0}.00 \n"
                                                             , BellevueVendCoke.PurchaseBalance);
                             BellevueVendCoke.DisplayMenu(); //Call the Display Menu Method.
                         }
                         else if (cokeAvailable == false) // If coke is not available.
                         {
                             Console.Clear();
                             Console.Write("Sorry, the machine is empty, enter an R to get your money back. \n");
                             BellevueVendCoke.DisplayMenu();
                         }
                         else // None of the conditions apply.
                         {   Console.Clear();
                             Console.Write("Sorry, you have to insert more money. \n");
                             BellevueVendCoke.DisplayMenu();
                         }
                         break;
                     case "R": // Get your money back.
                         Console.Clear();
                         fundsAvailable = BellevueVendCoke.GiveRefund(customerAmt); // Call the GiveRefund method.
                         if (fundsAvailable == true) // If funds are available allow the refund.
                         {
                             Console.Write("Here is your ${0}.00 \n", BellevueVendCoke.Balance);
                             BellevueVendCoke.DisplayMenu();
                             BellevueVendCoke.ZeroWithdrawlAcct(); // Zero the account upon refund.
                         }
                         else // You have no money in the machine.
                         {
                             Console.Clear();
                             Console.Write("Sorry, you need to insert a dollar. \n");
                             BellevueVendCoke.DisplayMenu();
                         }
                         break;

                      default:
                          Console.Write("Invalid selection. \n");
                          BellevueVendCoke.DisplayMenu();
                          break;
                } //End of switch.
                customerCommand = Console.ReadLine();
                customerCommand = customerCommand.ToUpper(); // Allow user to type upper or lower case.
            } //End of while.
            Console.Write("\n Thank you for purchasing our products." +
                          "\n Please press any key to end this program. ");
            Console.ReadKey(); // Pause before ending.
            Console.Clear();
        }
    }
}
```

Class Library.cs

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ClassLibrary1
{
    public class ChangeVend
    {
        // Private class fields to be used.
        private int _AccountBalance;
        private int _cokeBalance;


        // Display Menu Method.
        public void DisplayMenu()

        {   // Shows the menu using an array with strings.
            Console.ForegroundColor = ConsoleColor.White; string[] array = new string[5] {
                "\n Please type: P to insert a dollar,",
                      "\n\t      B for Buy a coke,",
                      "\n\t      R for Refund, or",
                      "\n\t      Q to Quit.\n",
                      "\n\t      Select your option: " };
            for (int i = 0; i < array.Length; ++i) { Console.Write(array[i]); }

        }

        // Constructor, starts the account balance at zero, coke balance to five.
        public ChangeVend()
        {
            _AccountBalance = 0;
            _cokeBalance = 5;
        }

        //=======================Regulates the coke================================
        // Check the coke balance. (Property)
        public int CokeBalance
        {
            get
            {
                return _cokeBalance;
            }
        }
        // Check to see if there is no coke in the machine. (Purchased 5 times)
        public bool BuyCoke(int coke)   // Method to allow coke Purchasing.
        {
            bool ok = true;
            if (_cokeBalance < 1) // If condition is true, there is no coke left.
            {
                coke = 0;
                ok = false;
            }
            else // There is still coke in the machine.
            {
                coke = 1; //Sets coke to minus 1 each time.
                _cokeBalance = _cokeBalance - coke; // Calculate how many cokes are left.
                Dispense(coke);
            }
            return ok;
        }


        //==========================Regulates the Money==================================
        // Get the money balance from the machine. (Property)
        public int PurchaseBalance
        {
            get
            {
                return _AccountBalance;
            }
        }
        // Check if there is any money in the machine.
        public bool CheckPurchase(int amount)   // Method to allow Purchasing.
        {
            bool ok = true;
            if (_AccountBalance < 1 || _cokeBalance == 0) // Check to see if there is no money or no coke.
            {
                ok = false;
            }
            else
            {
                _AccountBalance = _AccountBalance - amount; // Calculate: Balance minus amount. (Note: amount = 1)
                Dispense(amount);
            }
            return ok;
        }

        //Returns the money balance value.
        public int Balance
        {
            get
            {
                return _AccountBalance;
            }
        }

        //Deposit Money.
        public void AcceptCash(int amount)

        {
            _AccountBalance = _AccountBalance + amount; // Accept a deposit of 1 dollar each time.

        }

       // See if there is any money in the account, if so get the refund.
        public bool GiveRefund(int amount)   // Allow a full refund of the account balance.
        {
            bool ok = true;
            if (_AccountBalance < amount)
            {
                ok = false;
            }
            else
            {
                amount = 0; // Just get the account balance without amount.
                _AccountBalance = _AccountBalance - amount;
                Dispense(amount);
            }
            return ok;
        }

        // Make the account balance back to zero upon refund.
        public void ZeroWithdrawlAcct()
        {
            _AccountBalance = 0;
        }

        // Dispenses the cash and the coke out.
        private void Dispense(int cashOut) // private method can only be seen from inside this class
        {

        }
    }
}
```