Vending Machine – 30131623

Introduction

In this project, I will be creating a vending machine through C# windows application (a graphical user interface-based program).

Design



I have created my design based off a physical vending machine, with the snacks/drinks positioned in rows, and on the side of the “unit” there is a keypad with a screen that will output the numbers entered to confirm choices and display the price.  
Beneath the keypad is a coin slot, this is where the money on the side would be dragged to, for the user to pay through drag and drop.

The document this was created on was lost due to technical issues and it will be reduced to a screenshot.



Testing

Testing has been done where applicable, this is because how the vending machine was created it used repeated code and classes which also made it easier to create as well as test.

|  |
| --- |
| Test Case: 1 |
| Expected Outcome – Adds 103 (DR Pepper to the item list) |
| Outputs insufficient funds/invalid choice. Fix found. It was after a selection was made already and the write for the textbox adds a space which voids the request. Overwriting the choice textbox was done with txt\_choice.Text = “ “ Made it txt\_choice.Text = ““ |

|  |
| --- |
| Test Case: 2 |
| Expected Outcome – Outputs messagebox with Change: # |
| Outputs messagebox titled the change, saying “Change”.  x  Was previously MessageBox.Show(“Change: “, total.ToString()); |

|  |
| --- |
| Test Case 3: Add cheezits, changed to function |
| Expected output: Cheezits and the price is added to the list, price added to total |
| Outcome: As expected |

|  |
| --- |
| Test Case 4: Drag coin when checkout not confirmed |
| Expected Outcome: Denies money input, provides messagebox |
| Actual Outcome: As expected |

|  |
| --- |
| Test Case 5: Drag coin when checkout confirmed, complete payment |
| Expected Outcome: Transaction complete, change output if any, clears list |
| Actual Outcome: As Expected, list clears not shown in screenshot |

|  |
| --- |
| Test Case 6: Add MnMs |
| Expected Outcome: MnMs added to list |
| Actual Outcome: Price increase not added to list Found to be because of the reset after completing purchase.  Fix – in the reset after purchase, chhanged list\_items.Clear() to list\_items.Items.Clear() to only remove items, same code used for the rest of the products. Test applies to all |

|  |
| --- |
| Test Case 7: Add £1 in to order above £1 |
| Expected Outcome: Minus £1 from total |
| Actual Outcome: As expected, same code used for all money adding, test applies to all |

|  |
| --- |
| Test Case 8: Remove item |
| Expected Outcome: Removes item, reduces total |
| Actual Outcome: Removed item, reduces total, will change too clear text |

|  |
| --- |
| Test Case: 9 - Cancel order |
| Expected Outcome: Removes all items, and sets cost to 0, after confirmation |
| Actual Outcome: As expected |

Code

using Microsoft.VisualBasic;

using System.Diagnostics.Eventing.Reader;

using System.Security.Cryptography;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace VendingMachine\_30131623

{

public partial class Form1 : Form

{

public class Product

{

public string? ProductName;

public int Product\_no;

public double Product\_price;

}

Product CheezIts;

Product SkittlesS;

Product DrPepper;

Product MnMs;

bool checkout = false;

public Form1()

{

InitializeComponent();

CheezIts = new Product

{

ProductName = "CheezIts",

Product\_no = 101,

Product\_price = 2.00

};

SkittlesS = new Product // Object creation, sets product number, price and name

{

ProductName = "Skittles Sour",

Product\_no = 102,

Product\_price = 2.00

};

DrPepper = new Product

{

ProductName = "Dr Pepper",

Product\_no = 103,

Product\_price = 1.00

};

MnMs = new Product

{

ProductName = "M&Ms",

Product\_no = 104,

Product\_price = 2.50

};

Money\_in.Text = "0.00";

}

private void AddProduct(Product product)

{

AddProduct(product, product.ProductName);

}

private void RemoveProduct(Product product)

{

RemoveProduct(product, product.ProductName);

}

void AddProduct(Product product, string productName) // Passes through produce detaills (from object)

{ // Adds object to order list

double totalCost = 0;

double.TryParse(Money\_in.Text, out totalCost);

totalCost += product.Product\_price;

Money\_in.Text = totalCost.ToString("0.00");

productName = product.ProductName;

ListViewItem item = new ListViewItem(product.ProductName);

item.SubItems.Add(product.Product\_price.ToString("0.00"));

list\_items.Items.Add(item);

txt\_Choice.Text = "";

}

void RemoveProduct(Product product, string productName)

{

double totalCost = 0;

double.TryParse(Money\_in.Text, out totalCost);

for (int i = list\_items.Items.Count - 1; i >= 0; i--)

{

if (list\_items.Items[i].Text == productName)

{

list\_items.Items.RemoveAt(i);

totalCost -= product.Product\_price;

}

}

Money\_in.Text = totalCost.ToString("0.00");

txt\_Choice.Text = "";

}

private void Form1\_Load(object sender, EventArgs e)

{

one\_pence.Tag = "0.01"; // Provides tag for input to money in.

two\_pence.Tag = "0.02";

five\_pence.Tag = "0.05";

ten\_pence.Tag = "0.10";

twenty\_pence.Tag = "0.20";

fifty\_pence.Tag = "0.50";

one\_pound.Tag = "1.00";

coinslot.AllowDrop = true; // Allows images to be dropped

list\_items.View = View.Details;

list\_items.Columns.Add("Item", 120);

list\_items.Columns.Add("Price", 60);

}

#region dragndrop

private void one\_pence\_MouseDown(object sender, MouseEventArgs e)

{

if (sender is PictureBox pb && pb.Tag != null) // Lets you drag each coin to be dropped at the coinslot, amount is wwritten to the the "screen"

{

\_ = DoDragDrop(data: pb.Tag.ToString(), DragDropEffects.Copy);

}

}

private void two\_pence\_MouseDown(object sender, MouseEventArgs e)

{

if (sender is PictureBox pb && pb.Tag != null)

{

\_ = DoDragDrop(data: pb.Tag.ToString(), DragDropEffects.Copy);

}

}

private void five\_pence\_MouseDown(object sender, MouseEventArgs e)

{

if (sender is PictureBox pb && pb.Tag != null)

{

\_ = DoDragDrop(data: pb.Tag.ToString(), DragDropEffects.Copy);

}

}

private void ten\_pence\_MouseDown(object sender, MouseEventArgs e) // The drag and drop to insert money into the machine

{

if (sender is PictureBox pb && pb.Tag != null)

{

\_ = DoDragDrop(data: pb.Tag.ToString(), DragDropEffects.Copy);

}

}

private void twenty\_pence\_MouseDown(object sender, MouseEventArgs e)

{

if (sender is PictureBox pb && pb.Tag != null)

{

\_ = DoDragDrop(data: pb.Tag.ToString(), DragDropEffects.Copy);

}

}

private void fifty\_pence\_MouseDown(object sender, MouseEventArgs e)

{

if (sender is PictureBox pb && pb.Tag != null)

{

\_ = DoDragDrop(data: pb.Tag.ToString(), DragDropEffects.Copy);

}

}

private void one\_pound\_MouseDown(object sender, MouseEventArgs e)

{

if (sender is PictureBox pb && pb.Tag != null)

{

\_ = DoDragDrop(data: pb.Tag.ToString(), DragDropEffects.Copy);

}

}

double totalInserted = 0.0;

private void coinslot\_DragEnter(object sender, DragEventArgs e)

{

if (e.Data.GetDataPresent(DataFormats.Text))

e.Effect = DragDropEffects.Copy;

}

private void coinslot\_DragDrop(object sender, DragEventArgs e)

{

string data = (string)e.Data.GetData(DataFormats.Text);

if (double.TryParse(data, out double coinValue) && checkout == true)

{

totalInserted += coinValue;

double inputValue = Convert.ToDouble(Money\_in.Text); // adds up money from the inserted coins

inputValue -= totalInserted;

Money\_in.Text = inputValue.ToString();

if (Convert.ToDouble(Money\_in.Text) <= 0)

{

double change;

change = System.Math.Abs(Convert.ToDouble(Money\_in.Text));

MessageBox.Show("Transaction Complete. Change: £" + change.ToString()); // outputs change is money insterted is

checkout = false; // above required

list\_items.Items.Clear();

Money\_in.Text = "0.00";

}

}

else if (double.TryParse(data, out coinValue) && checkout == false)

{

totalInserted += coinValue;

MessageBox.Show("Confirm order first. Change: £" + totalInserted.ToString());

}

}

#endregion

#region Number input

private void btn\_one\_Click(object sender, EventArgs e) // sets up keypad to displayy item numbers on screen

{

txt\_Choice.Text = txt\_Choice.Text + "1";

}

private void btn\_two\_Click(object sender, EventArgs e)

{

txt\_Choice.Text = txt\_Choice.Text + "2";

}

private void btn\_three\_Click(object sender, EventArgs e)

{

txt\_Choice.Text = txt\_Choice.Text + "3";

}

private void btn\_four\_Click(object sender, EventArgs e)

{

txt\_Choice.Text = txt\_Choice.Text + "4";

}

private void btn\_five\_Click(object sender, EventArgs e)

{

txt\_Choice.Text = txt\_Choice.Text + "5";

}

private void btn\_six\_Click(object sender, EventArgs e) // Each takes input from the button and saves input to choice textbox

{

txt\_Choice.Text = txt\_Choice.Text + "6";

}

private void btn\_seven\_Click(object sender, EventArgs e)

{

txt\_Choice.Text = txt\_Choice.Text + "7";

}

private void btn\_eight\_Click(object sender, EventArgs e)

{

txt\_Choice.Text = txt\_Choice.Text + "8";

}

private void btn\_nine\_Click(object sender, EventArgs e)

{

txt\_Choice.Text = txt\_Choice.Text + "9";

}

private void btn\_zero\_Click(object sender, EventArgs e)

{

txt\_Choice.Text = txt\_Choice.Text + "0";

}

#endregion

double value;

string snack;

private void btn\_enter\_Click(object sender, EventArgs e)

{

double totalCost;

if (int.Parse(txt\_Choice.Text) == CheezIts.Product\_no && checkout == false)

{

AddProduct(CheezIts);

}

else if (int.Parse(txt\_Choice.Text) == SkittlesS.Product\_no && checkout == false)

{

AddProduct(SkittlesS);

}

else if (int.Parse(txt\_Choice.Text) == DrPepper.Product\_no && checkout == false)

{

AddProduct(DrPepper);

}

else if (int.Parse(txt\_Choice.Text) == MnMs.Product\_no && checkout == false)

{

AddProduct(MnMs);

}

else

{ // If statement for the cases that a number is entered that does not

MessageBox.Show("Invalid Choice"); // have a product linked

txt\_Choice.Text = "";

}

if (checkout == true)

{ // If checkout has been chosen, user told to insert

MessageBox.Show("Insert coins, total: £", Money\_in.Text); // money instead of adding more products

}

}

private void btn\_cancel\_Click(object sender, EventArgs e)

{

txt\_Choice.Text = "";

}

private void btn\_checkout\_Click(object sender, EventArgs e)

{

checkout = true;

}

private void btn\_void\_Click(object sender, EventArgs e)

{

if (int.Parse(txt\_Choice.Text) == CheezIts.Product\_no && checkout == false)

{

RemoveProduct(CheezIts);

}

else if (int.Parse(txt\_Choice.Text) == SkittlesS.Product\_no && checkout == false)

{ // removal of products from order

RemoveProduct(SkittlesS);

}

else if (int.Parse(txt\_Choice.Text) == DrPepper.Product\_no && checkout == false)

{

RemoveProduct(DrPepper);

}

else if (int.Parse(txt\_Choice.Text) == MnMs.Product\_no && checkout == false)

{

RemoveProduct(MnMs);

}

else

{ // If statement for the cases that a number is entered that does not

MessageBox.Show("Invalid Choice"); // have a product linked

txt\_Choice.Text = "";

}

if (checkout == true)

{ // If checkout has been chosen, user told to insert

MessageBox.Show("Insert coins, total: £", Money\_in.Text); // money instead of adding more products

}

}

private void CheezIt\_Click(object sender, EventArgs e)

{

}

private void coinslot\_Click(object sender, EventArgs e)

{

}

private void btn\_cancelorder\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show("Cancel Order?", "Cancel Confirmation",

MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

list\_items.Items.Clear();

Money\_in.Text = "0.00";

}

References

BillWagner (2025). *Classes*. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/types/classes>.