**Vending Machine Design – Jordan Foreman, 17033128**

**Contents**

--Form Design (1)

--Code Design (3)

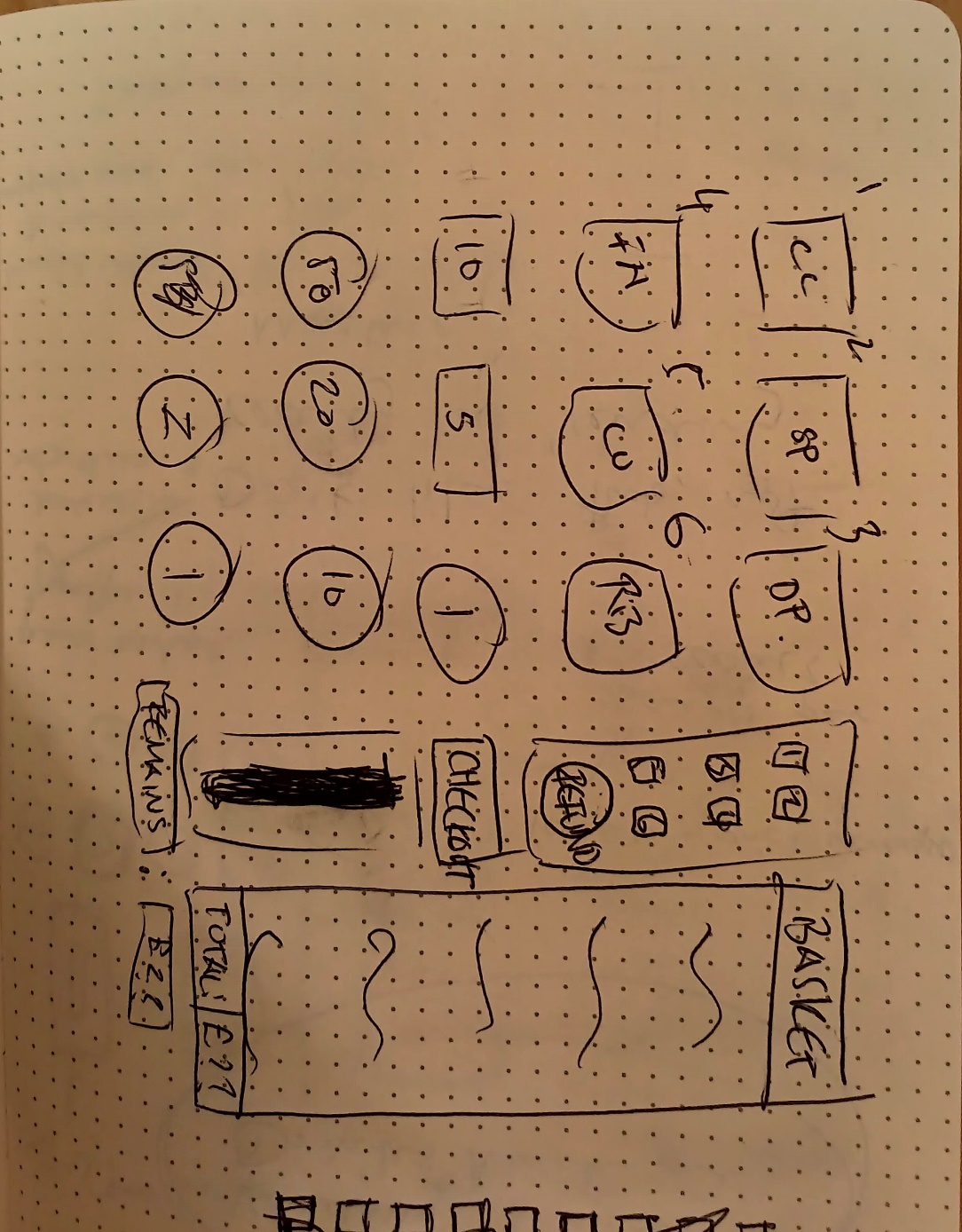
--Testing Evidence ()

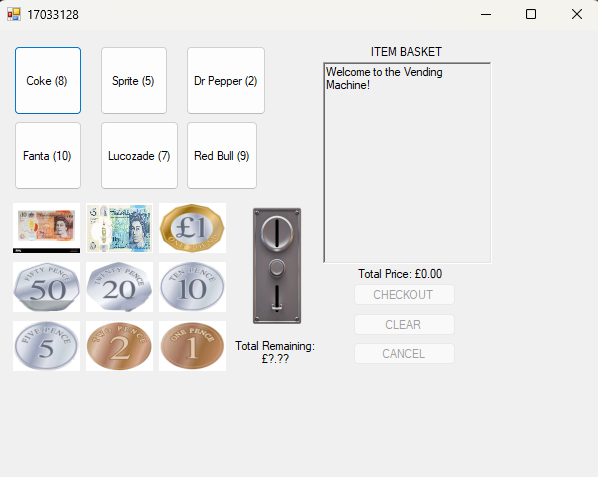
--Source Code ()

**--Form Design**

For the design, it was initially planned to be a fully according to brief vending machine, with six options to choose from (a reasonable two above the minimum requirement), allowing for the attractive symmetry layout of a 3x2 grid; which linked with the design choice of the 9x9 grid for the money/checkout section. Given the brief constraint of a basket system, a larger £10 note and £5 note needed to be added as well as the coins for quality of life. Removing the £2 coin from the money choices allowed the 9x9 grid including the notes and coins together.

After considering the grid sizes, a design sketch was made (shown below).





The design sketch ended up making it fairly close to the final product (shown above), with only a few minor changes occurring:

A close up of a paper

Description automatically generatedThe Vending Stock Sidebar:

This was removed entirely due to the functionality being better suited elsewhere. Instead of number labels on the vending stock choices and pressable separate buttons, the user was simply able to press the vending stocks button e.g., they could simply press “coke” instead of reading “coke: btn1” and pressing 1 on the sidebar. The system was needlessly complicated. The refund button was later renamed to “cancel” to match the wording of the brief and move amongst the other buttons as the sidebar no longer existed in production.

A close up of a board

Description automatically generatedThe Lower Basket and Buttons Section:

Before production started, the clear and cancel functionality was forgotten about until reading the brief at a much later (last minute) date, leading to a lot of production issues. Considering the new two buttons, the lower area of the basket was shuffled around to group all the buttons together for aesthetic purposes.

**--Code Design**

The code was designed in a format of four classes:

*public partial class {FORM}:*

*//application initalisation*

*//checkout enabler and disabler*

*//VendingStock quantity and button label handling*

*//clear button and checking functions*

*//button click events*

*//checkout section events*

*public class VendingStock:*

*//new stock item initialization*

*//quantity remaining getter*

*public class basketHandler:*

*//initialization of the dictionary containing the value of the quantity with a key of the price*

*//a clear basket function*

*//total price getter*

*//basket dictionary getter*

*public class Checkout:*

*//initialization of money variables, with variable names corresponding to the image names on the form, and values of their monetary equivalent decimal*

*//translator of image dragdrop event string dropped to the equivalent decimal in the switch case*

*//”add” money function that removed money from the total remaining*

*//total remaining getter and setter*

This separation proved very useful to the mental organization of where everything could be found within the program, as each class represents a category or section of work to be done. Each class was developed as the previous one needed linking, then allowing quick understanding of where whatever was being done was located. If development was being done on the checkout section, a large majority of functionality could be found within checkout, and so on.

**--Testing Evidence**

When testing the code base, a couple of different methods were employed. Mostly used were Console.Writeline statements within functions to determine what was being called properly and what they were receiving/returning.

A screen shot of a computer program

Description automatically generated

*(Above) An example of the Console.WriteLine being used within the checkoutButton\_MouseClick event function*

*A screen shot of a computer program

Description automatically generated*

*(Above) An example of the Console.WriteLine being used within the coinSlotImage\_DragDrop event function*

In the specific examples above, these outputs allowed the visualization of the value within the totalRemaining variable at multiple stages of it’s manipulation. Errors in the logic, knowing that it should have copied the value visible in totalPrice in the first instance, then minus the money inputted in the second instance; it was easy to tell where the program’s logic was faulted.

The large majority of testing however was trial and error, seeing where issues such as negative quantities arose and changing relevant function logic or creating entire new systems to mitigate errors. For example, the clear button would consistently refund all quantity from the basket whenever clicked, leading to a surplus that didn’t exist before. In order to fix it, the clear button was disabled when totalPrice == 0, as the variable was set to zero when the clear button was pressed; leading to a easy and working fix.

**--Source Code**

A (hopefully functional) markdown file has been included for readability with the added benefit of copy and paste functionality. Additionally, the code is simply pasted below straight from the IDE with no further edits.



**using System;**

**using System.Collections.Generic;**

**using System.ComponentModel;**

**using System.Data;**

**using System.Data.SqlTypes;**

**using System.Diagnostics;**

**using System.Drawing;**

**using System.Drawing.Printing;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**using System.Windows.Forms;**

**using System.Xml.Linq;**

**namespace vendingMachine**

**{**

**public partial class SN17033128 : Form**

**{**

**private basketHandler basket;**

**private VendingStock coke;**

**private VendingStock sprite;**

**private VendingStock drPepper;**

**private VendingStock fanta;**

**private VendingStock lucozade;**

**private VendingStock redBull;**

**Checkout checkout = new Checkout();**

**public SN17033128()**

**{**

**InitializeComponent();**

**coke = new VendingStock("Coke", 2.50m, 8);**

**sprite = new VendingStock("Sprite", 1.95m, 5);**

**drPepper = new VendingStock("Dr Pepper", 2.10m, 2);**

**fanta = new VendingStock("Fanta", 1.90m, 10);**

**lucozade = new VendingStock("Lucozade", 2.20m, 7);**

**redBull = new VendingStock("Red Bull", 2.60m, 9);**

**basket = new basketHandler();**

**UpdateButtonLabels();**

**clearButton\_Check();**

**checkoutSectionDisabler();**

**coinSlotImage.AllowDrop = true;**

**}**

**/\***

**Two functions that deal with enabling and disabling the checkout section, opposing the vending section**

**\*/**

**public void checkoutSectionEnabler()**

**{**

**//checkout enabled**

**money1000.Enabled = true;**

**money500.Enabled = true;**

**money100.Enabled = true;**

**money50.Enabled = true;**

**money20.Enabled = true;**

**money10.Enabled = true;**

**money5.Enabled = true;**

**money2.Enabled = true;**

**money1.Enabled = true;**

**coinSlotImage.Enabled = true;**

**cancelButton.Enabled = true;**

**//vending enabled**

**cokeButton.Enabled = false;**

**spriteButton.Enabled = false;**

**drPepperButton.Enabled = false;**

**fantaButton.Enabled = false;**

**lucozadeButton.Enabled = false;**

**redBullButton.Enabled = false;**

**clearButton.Enabled = false;**

**checkoutButton.Enabled = false;**

**}**

**public void checkoutSectionDisabler()**

**{**

**//checkout enabled**

**money1000.Enabled = false;**

**money500.Enabled = false;**

**money100.Enabled = false;**

**money50.Enabled = false;**

**money20.Enabled = false;**

**money10.Enabled = false;**

**money5.Enabled = false;**

**money2.Enabled = false;**

**money1.Enabled = false;**

**coinSlotImage.Enabled = false;**

**cancelButton.Enabled = false;**

**//vending enabled**

**cokeButton.Enabled = true;**

**spriteButton.Enabled = true;**

**drPepperButton.Enabled = true;**

**fantaButton.Enabled = true;**

**lucozadeButton.Enabled = true;**

**redBullButton.Enabled = true;**

**}**

**/\***

**Group of functions that handle the button clicks for the vending machine buttons and basket logic**

**\*/**

**private void buttonClickQuantityChange(VendingStock vendingStock, Button button)**

**{**

**if (vendingStock.getQuantityRemaining() > 1)**

**{**

**vendingStock.quantityRemaining -= 1;**

**UpdateButtonText(vendingStock);**

**}**

**else if (vendingStock.getQuantityRemaining() == 1)**

**{**

**vendingStock.quantityRemaining -= 1;**

**UpdateButtonText(vendingStock);**

**button.Enabled = false;**

**}**

**}**

**private void UpdateTotalPriceLabel()**

**{**

**totalPriceLabelUpdateable.Text = $"Total Price: £{basket.GetTotalPrice():0.00}";**

**}**

**private void UpdateItemBasketTextBox()**

**{**

**itemBasketRichTextBox.Clear();**

**foreach (var item in basket.GetBasketItems())**

**{**

**itemBasketRichTextBox.AppendText($"{item.Key} - £{item.Value.price:0.00} - {item.Value.quantity}\n");**

**}**

**}**

**private void UpdateButtonText(VendingStock vendingStock)**

**{**

**if (vendingStock == coke)**

**{**

**cokeButton.Text = $"Coke ({coke.getQuantityRemaining()})";**

**}**

**else if (vendingStock == sprite)**

**{**

**spriteButton.Text = $"Sprite ({sprite.getQuantityRemaining()})";**

**}**

**else if (vendingStock == drPepper)**

**{**

**drPepperButton.Text = $"Dr Pepper ({drPepper.getQuantityRemaining()})";**

**}**

**else if (vendingStock == fanta)**

**{**

**fantaButton.Text = $"Fanta ({fanta.getQuantityRemaining()})";**

**}**

**else if (vendingStock == lucozade)**

**{**

**lucozadeButton.Text = $"Lucozade ({lucozade.getQuantityRemaining()})";**

**}**

**else if (vendingStock == redBull)**

**{**

**redBullButton.Text = $"Red Bull ({redBull.getQuantityRemaining()})";**

**}**

**}**

**private void UpdateButtonLabels()**

**{**

**cokeButton.Enabled = true;**

**cokeButton.Text = $"Coke ({coke.getQuantityRemaining()})";**

**spriteButton.Enabled = true;**

**spriteButton.Text = $"Sprite ({sprite.getQuantityRemaining()})";**

**drPepperButton.Enabled = true;**

**drPepperButton.Text = $"Dr Pepper ({drPepper.getQuantityRemaining()})";**

**fantaButton.Enabled = true;**

**fantaButton.Text = $"Fanta ({fanta.getQuantityRemaining()})";**

**lucozadeButton.Enabled = true;**

**lucozadeButton.Text = $"Lucozade ({lucozade.getQuantityRemaining()})";**

**redBullButton.Enabled = true;**

**redBullButton.Text = $"Red Bull ({redBull.getQuantityRemaining()})";**

**}**

**private void itemBasketTextBox\_TextChanged(object sender, EventArgs e)**

**{**

**UpdateItemBasketTextBox();**

**}**

**private void clearButton\_Check()**

**{**

**if (totalPriceLabelUpdateable.Text == $"Total Price: £0.00")**

**{**

**clearButton.Enabled = false;**

**checkoutButton.Enabled = false;**

**}**

**else**

**{**

**clearButton.Enabled = true;**

**checkoutButton.Enabled = true;**

**}**

**}**

**private void quantityCheck()**

**{**

**if (coke.getQuantityRemaining() == 0)**

**{**

**cokeButton.Enabled = false;**

**}**

**if (sprite.getQuantityRemaining() == 0)**

**{**

**spriteButton.Enabled = false;**

**}**

**if (drPepper.getQuantityRemaining() == 0)**

**{**

**drPepperButton.Enabled = false;**

**}**

**if (fanta.getQuantityRemaining() == 0)**

**{**

**fantaButton.Enabled = false;**

**}**

**if (lucozade.getQuantityRemaining() == 0)**

**{**

**lucozadeButton.Enabled = false;**

**}**

**if (redBull.getQuantityRemaining() == 0)**

**{**

**redBullButton.Enabled = false;**

**}**

**}**

**public void returnToVending()**

**{**

**// Clear the basket and add the quantity remaining back to the VendingMachine objects**

**foreach (var item in basket.GetBasketItems())**

**{**

**string itemName = item.Key;**

**int quantity = item.Value.quantity;**

**if (itemName == "Coke")**

**{**

**coke.quantityRemaining += quantity;**

**}**

**else if (itemName == "Sprite")**

**{**

**sprite.quantityRemaining += quantity;**

**}**

**else if (itemName == "Dr Pepper")**

**{**

**drPepper.quantityRemaining += quantity;**

**}**

**else if (itemName == "Fanta")**

**{**

**fanta.quantityRemaining += quantity;**

**}**

**else if (itemName == "Lucozade")**

**{**

**lucozade.quantityRemaining += quantity;**

**}**

**else if (itemName == "Red Bull")**

**{**

**redBull.quantityRemaining += quantity;**

**}**

**}**

**}**

**public void clearButton\_Click(object sender, EventArgs e)**

**{**

**returnToVending();**

**basket.clearBasket();**

**UpdateTotalPriceLabel();**

**UpdateItemBasketTextBox();**

**UpdateButtonLabels();**

**clearButton\_Check();**

**}**

**/\***

**All of the button\_click events for the vending machine buttons**

**\*/**

**private void cokeButton\_Click(object sender, EventArgs e)**

**{**

**buttonClickQuantityChange(coke, cokeButton);**

**basket.addToBasket("Coke", 2.50m);**

**UpdateTotalPriceLabel();**

**UpdateItemBasketTextBox();**

**clearButton\_Check();**

**}**

**private void spriteButton\_Click(object sender, EventArgs e)**

**{**

**buttonClickQuantityChange(sprite, spriteButton);**

**basket.addToBasket("Sprite", 1.95m);**

**UpdateTotalPriceLabel();**

**UpdateItemBasketTextBox();**

**clearButton\_Check();**

**}**

**private void drPepperButton\_Click(object sender, EventArgs e)**

**{**

**buttonClickQuantityChange(drPepper, drPepperButton);**

**basket.addToBasket("Dr Pepper", 2.10m);**

**UpdateTotalPriceLabel();**

**UpdateItemBasketTextBox();**

**clearButton\_Check();**

**}**

**private void fantaButton\_Click(object sender, EventArgs e)**

**{**

**buttonClickQuantityChange(fanta, fantaButton);**

**basket.addToBasket("Fanta", 1.90m);**

**UpdateTotalPriceLabel();**

**UpdateItemBasketTextBox();**

**clearButton\_Check();**

**}**

**private void lucozadeButton\_Click(object sender, EventArgs e)**

**{**

**buttonClickQuantityChange(lucozade, lucozadeButton);**

**basket.addToBasket("Lucozade", 2.20m);**

**UpdateTotalPriceLabel();**

**UpdateItemBasketTextBox();**

**clearButton\_Check();**

**}**

**private void redBullButton\_Click(object sender, EventArgs e)**

**{**

**buttonClickQuantityChange(redBull, redBullButton);**

**basket.addToBasket("Red Bull", 2.60m);**

**UpdateTotalPriceLabel();**

**UpdateItemBasketTextBox();**

**clearButton\_Check();**

**}**

**private void checkoutButton\_MouseClick(object sender, MouseEventArgs e)**

**{**

**totalPriceLabelUpdateable.Text = $"Total Price: £0.00";**

**checkoutSectionEnabler();**

**checkout.setTotalRemaining(basket.GetTotalPrice());**

**totalRemainingLabel.Text = $"Total Remaining: £{checkout.getTotalRemaining()}";**

**Console.WriteLine("Checkout button pressed. totalRemaining Variable contains:" + checkout.getTotalRemaining());**

**}**

**/\***

**Section dealing with all the money drag and drops, using the checkout class for after the checkout button has been pressed**

**\*/**

**//coin slot drag and drop w/ totalRemainingLabal changes**

**private void coinSlotImage\_DragDrop(object sender, DragEventArgs e)**

**{**

**checkout.moneyTranslate(e.Data.GetData(DataFormats.Text).ToString());**

**totalRemainingLabel.Text = $"Total Remaining: £{checkout.getTotalRemaining()}";**

**Console.WriteLine("Dropped in coin slot. totalRemaining Variable contains:" + checkout.getTotalRemaining());**

**allPaidUpChecker();**

**}**

**private void coinSlotImage\_DragEnter(object sender, DragEventArgs e)**

**{**

**e.Effect = DragDropEffects.Copy;**

**}**

**void allPaidUpChecker()**

**{**

**if (checkout.getTotalRemaining() == 0)**

**{**

**MessageBox.Show("All paid up! Enjoy your drinks!");**

**ResetApplication();**

**}**

**}**

**private void cancelButton\_Click(object sender, EventArgs e)**

**{**

**DialogResult result = MessageBox.Show("Are you sure you want to cancel?", "Cancel Confirmation", MessageBoxButtons.YesNo);**

**if (result == DialogResult.Yes)**

**{**

**returnToVending();**

**ResetApplication();**

**totalRemainingLabel.Text = $"Total Remaining: £{checkout.getTotalRemaining()}";**

**}**

**}**

**private void ResetApplication()**

**{**

**checkout.setTotalRemaining(0.00m);**

**basket.clearBasket();**

**UpdateTotalPriceLabel();**

**UpdateItemBasketTextBox();**

**UpdateButtonLabels();**

**checkoutSectionDisabler();**

**clearButton\_Check();**

**quantityCheck();**

**}**

**//all money drag and drops**

**private void money1000\_MouseDown(object sender, MouseEventArgs e)**

**{**

**money1000.DoDragDrop("money1000", DragDropEffects.Copy);**

**}**

**private void money500\_MouseDown(object sender, MouseEventArgs e)**

**{**

**money500.DoDragDrop("money500", DragDropEffects.Copy);**

**}**

**private void money100\_MouseDown(object sender, MouseEventArgs e)**

**{**

**money100.DoDragDrop("money100", DragDropEffects.Copy);**

**}**

**private void money50\_MouseDown(object sender, MouseEventArgs e)**

**{**

**money50.DoDragDrop("money50", DragDropEffects.Copy);**

**}**

**private void money20\_MouseDown(object sender, MouseEventArgs e)**

**{**

**money20.DoDragDrop("money20", DragDropEffects.Copy);**

**}**

**private void money10\_MouseDown(object sender, MouseEventArgs e)**

**{**

**money10.DoDragDrop("money10", DragDropEffects.Copy);**

**}**

**private void money5\_MouseDown(object sender, MouseEventArgs e)**

**{**

**money5.DoDragDrop("money5", DragDropEffects.Copy);**

**}**

**private void money2\_MouseDown(object sender, MouseEventArgs e)**

**{**

**money2.DoDragDrop("money2", DragDropEffects.Copy);**

**}**

**private void money1\_MouseDown(object sender, MouseEventArgs e)**

**{**

**money1.DoDragDrop("money1", DragDropEffects.Copy);**

**}**

**}**

**/\***

**Class dealing with the stock of the vending machine**

**\*/**

**public class VendingStock**

**{**

**string name = " ";**

**decimal price = 0.0m;**

**public int quantityRemaining = 0;**

**public VendingStock(string name, decimal price, int quantityRemaining)**

**{**

**this.name = name;**

**this.price = price;**

**this.quantityRemaining = quantityRemaining;**

**}**

**public int getQuantityRemaining()**

**{**

**return quantityRemaining;**

**}**

**}**

**/\***

**Class dealing with the basket of the vending machine, separated for readability**

**\*/**

**public class basketHandler**

**{**

**decimal totalPrice = 0.0m;**

**Dictionary<string, (decimal price, int quantity)> basketItems = new Dictionary<string, (decimal price, int quantity)>();**

**public void addToBasket(string item, decimal price)**

**{**

**if (basketItems.ContainsKey(item))**

**{**

**basketItems[item] = (price, basketItems[item].quantity + 1);**

**}**

**else**

**{**

**basketItems[item] = (price, 1);**

**}**

**totalPrice += price;**

**}**

**public void clearBasket()**

**{**

**basketItems.Clear();**

**totalPrice = 0.0m;**

**}**

**public decimal GetTotalPrice()**

**{**

**return totalPrice;**

**}**

**public Dictionary<string, (decimal price, int quantity)> GetBasketItems()**

**{**

**return basketItems;**

**}**

**}**

**/\***

**Class dealing with the money in the vending machine**

**\*/**

**public class Checkout**

**{**

**decimal money1000 = 10.00m;**

**decimal money500 = 5.00m;**

**decimal money100 = 1.00m;**

**decimal money50 = 0.50m;**

**decimal money20 = 0.20m;**

**decimal money10 = 0.10m;**

**decimal money5 = 0.05m;**

**decimal money2 = 0.02m;**

**decimal money1 = 0.01m;**

**decimal totalRemaining = 0.0m;**

**public void moneyTranslate(string input)**

**{**

**switch (input)**

**{**

**case "money1000":**

**addMoney(money1000);**

**break;**

**case "money500":**

**addMoney(money500);**

**break;**

**case "money100":**

**addMoney(money100);**

**break;**

**case "money50":**

**addMoney(money50);**

**break;**

**case "money20":**

**addMoney(money20);**

**break;**

**case "money10":**

**addMoney(money10);**

**break;**

**case "money5":**

**addMoney(money5);**

**break;**

**case "money2":**

**addMoney(money2);**

**break;**

**case "money1":**

**addMoney(money1);**

**break;**

**}**

**}**

**//decrements the remaining money in the checkout section**

**public void addMoney(decimal money)**

**{**

**if (totalRemaining - money >= 0)**

**{**

**totalRemaining -= money;**

**}**

**else**

**{**

**MessageBox.Show($"That's too much! You only have £{totalRemaining} left to pay!");**

**}**

**}**

**public decimal setTotalRemaining(decimal total)**

**{**

**totalRemaining = total;**

**return totalRemaining;**

**}**

**public decimal getTotalRemaining()**

**{**

**return totalRemaining;**

**}**

**}**

**}**