|  |  |  |
| --- | --- | --- |
| **Object** | **Properties** | **Procedure** |
| buttonExit | Caption of “Exit”  Arial narrow text, 12pt size | End program |
| ButtonReset | Caption of “Reset”  Arial narrow text, 12pt size | Reset the number of each drink sold to 0 |
| ButtonReport | Caption of “Report”  Arial narrow text, 12pt size | Generate and display report of number of each drink sold |
| ButtonPriceList | Caption of “Price List”  Arial narrow text, 12pt size | Generate and display table of prices for each drink |
| ButtonOrder | Caption of “Order”  Arial narrow text, 12pt size | Display order interface |
| ButtonFlatWhite | Caption of “Flat White”  Arial narrow text, 36pt size | * Set DrinkType to “Flat White” * Display size option buttons “ButtonSizeSmall” “ButtonSizeLarge” |
| ButtonCappucino | Caption of “Cappucino”  Arial narrow text, 36pt size | * Set DrinkType to “Cappucino” * Display size option buttons “ButtonSizeSmall” “ButtonSizeLarge” |
| ButtonLongMac | Caption of “Long Mac”  Arial narrow text, 36pt size | * Set DrinkType to “Long Mac” * Display size option buttons “ButtonSizeSmall” “ButtonSizeLarge” |
| ButtonEspressoShot | Caption of “Espresso Shot”  Arial narrow text, 36pt size | * Set DrinkType to “Espresso Shot” * Display size option buttons “ButtonSizeSmall” “ButtonSizeLarge” |
| ButtonDrinkSizeSmall | Caption of “Small”  Arial narrow text, 36pt size | * Set DrinkSize to “S” * Display InputQuantity |
| ButtonDrinkSizeLarge | Caption of “Large”  Arial narrow text, 36pt size | * Set DrinkSize to “L” * Display InputQuantity |
| InputBoxDrinkQuantity | Default amount of “1” | * Input box for user to input quantity of drink * Display DrinkQuantity * Set drink DrinkQuantity to user inputted amount |
| ButtonDrinkQuantityDecrease | Caption of “-”  Arial narrow text, 36pt size | * Decrease DrinkQuantity by 1 * Refresh Displayed DrinkQuantity |
| ButtonDrinkQuantityIncrease | Caption of “+”  Arial narrow text, 36pt size | * Increase DrinkQuantity by 1 * Refresh Displayed DrinkQuantity |
| ButtonAddToOrder | Caption of “Add to Order”  Arial narrow text, 24pt size | * Increase orderQuantity by 1 i.e. counts the number of different drinks (different records) in the order * Calculate drinkOrderPrice using drinkType, drinkSize, drinkQuantity * Create record "drinkOrder" with drinkType, drinkSize, drinkQuantity, drinkOrderPrice * Refresh displayed drink orders * Calculate and refresh orderSubtotal, orderTotal * Reset drink variables |
| ButtonRemoveFromOrder | Caption of “Remove from Order”  Arial narrow text, 24pt size | Removes ???record??? from order ???entity??? |
| ButtonPrimaBeansYes | Caption of “Y”  Arial narrow text, 24pt size | Set PrimaBeansMember to True |
| ButtonPrimaBeansNo | Caption of “N”  Arial narrow text, 24pt size | Set PrimaBeansMember to False |
| ButtonCompleteOrder | Caption of “Complete Order”  Arial narrow text, 24pt size | Adds drinks to report entity  Resets order variables |

Graphical user interface, application, Teams

Description automatically generatedA picture containing table

Description automatically generated

Diagram, schematic

Description automatically generated

Pseudocode:

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ INITIALISATION OF VARIABLES ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

drinkPrice = 0

drinkType = ""

drinkSize = ""

drinkQuantity = 0

UserInputtedDrinkQuantity = 0

orderSubtotal = 0

orderTotal = 0

orderList = []

reportList = []

primaBeansMember = false

orderDiscount = 0

totalEspressoShot = 0

totalLongMac = 0

totalCappucino = 0

totalFlatWhite = 0

reportTotal = 0

reportDrinkType = 0

reportItemPrice = 0

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ BUTTONS ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Module exit(): #ButtonExit

    @End Program

End Module

Module reset(): #ButtonReset

    drinkPrice = 0

    drinkType = ""

    drinkSize = ""

    drinkQuantity = 0

    UserInputtedDrinkQuantity = 0

    orderSubtotal = 0

    orderTotal = 0

    orderList = []

    primaBeansMember = false

    orderDiscount = 0

End Module

Module PriceList(): #ButtonPriceList

    @Generate and display table of prices for each drink on menu

End Module

Module Report(): #ButtonReport

    output(reportInterface)

End Module

Module Order(): #ButtonOrder

    output(orderInterface)

End Module

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ CALCULATIONS ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Function calcDrinkPrice(): #Calculates drinkPrice from drinkType and drinkSize

    If drinkType = "Espresso Shot":

        return 3

    Else If: drinkType = "Long Mac":

        If drinkSize = "Small":

            return 4.6

        Else:

            return 5

    Else If: drinkType = "Cappucino":

        If drinkSize = "Small":

            return 4.5

        Else:

            return 5

    Else If: drinkType = "Flat White":

        If drinkSize = "Small":

            return 4.3

        Else:

            return 4.8

End Function

Module calcOrderTotal(): #Calculates orderTotal

    @

End Module

Function orderValidation(drinkType, drinkSize, drinkQuantity):

    If drinkType = "": #checks that a drink type has been selected

        output("Please select a drink type.") #outputs an error message If a drink type has not been selected

        return True #returns True If there is an error

    Else If: drinkSize = "": #checks that a drink type has been selected

        output("Please select a drink size.") #outputs an error message If a drink size has not been selected

        return True

    Else If: drinkQuantity <= 0 OR type(drinkQuantity) is not integer: #checks that the drink quantity is an integer and that drink quantity is greater than 0        output("Please input an integer quantity greater than 0.")

        return True

    Else:

        return False #returns False If there is not an error

    End If

End Function

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ORDER INTERFACE ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Module updateDrinkPrice():

    output(drinkPrice)

End Module

Module flatWhite(): #ButtonFlatWhite

    drinkType = "Flat White" #Sets drinkType to "Flat White"

End Module

Module cappucino(): #ButtonCappucino

    drinkType = "Cappucino" #Sets drinkType to "Cappucino"

End Module

Module longMac(): #ButtonLongMac

    drinkType = "Long Mac" #Sets drinkType to "Long Mac"

End Module

Module espressoShot(): #ButtonEspressoShot

    drinkType = "Espresso Shot" #Sets drinkType to "Espresso Shot"

End Module

Module drinkSizeSmall(): #ButtonDrinkSizeSmall

    drinkSize = "Small" #Sets drinkSize to "Small"

End Module

Module drinkSizeLarge(): #ButtonDrinkSizeLarge

    drinkSize = "Large" #Sets drinkSize to "Large"

End Module

Module drinkQuantity(): #InputBoxDrinkQuantity

    input(UserInputtedDrinkQuantity) #Sets drinkQuantity to user inputted value

    If ((drinkQuantityLengthValidation(UserInputtedDrinkQuantity)) == false):

        If ((drinkQuantityIntegerValidation(UserInputtedDrinkQuantity)) == false):

            If ((drinkQuantityParityValidation(UserInputtedDrinkQuantity)) == false):

                drinkQuantity = int(UserInputtedDrinkQuantity)

            End If

            Else:

                drinkQuantity = 0

                output(DrinkQuantityError) #If user inputted drink quantity is not an integer, outputs an error message

            End If

        End If

        Else:

            drinkQuantity = 0

            output(DrinkQuantityError) #If user inputted drink quantity is not an integer, outputs an error message

        End If

    End If

    Else:

        drinkQuantity = 0

        output(DrinkQuantityError) #If user inputted drink quantity is not an integer, outputs an error message

    End If

End Module

Module drinkQuantityDecrease(): #ButtonDrinkQuantityDecrease

    If drinkQuantity > 0: #Prevents drinkQuantity from being negative

        drinkQuantity -= 1 #Decreases drinkQuantity by 1

    End If

End Module

Module drinkQuantityIncrease(): #ButtonDrinkQuantityIncrease

    drinkQuantity += 1 #Increases drinkQuantity by 1

End Module

Module addToOrder(): #ButtonAddToOrder

    If ((orderValidation(drinkType, drinkSize, drinkQuantity)) == false):

        drinkPrice = calcDrinkPrice(drinkType,drinkSize)

        itemPrice = drinkQuantity \* drinkPrice

        orderItem = [drinkType, drinkSize, drinkQuantity, drinkPrice, itemPrice]; #creates new instance of drinkOrder class using current drink configuration

        append orderItem to orderList #adds the new order item to the order list

        orderPrice = 0;

        calcOrderSubtotal() #calculate and refresh orderSubtotal, orderTotal

        resetItem() #reset drink variables

End Module

Module PrimaBeansYes():

    PrimaBeansMember = True

    updateOrderTotal()

End Module

Module PrimaBeansNo():

    PrimaBeansMember = False

    updateOrderTotal()

End Module

Function drinkQuantityValidation(UserInputtedDrinkQuantity):

    error = drinkQuantityLengthValidation(UserInputtedDrinkQuantity)

    error = drinkQuantityIntegerValidation(UserInputtedDrinkQuantity)

    error = drinkQuantityParityValidation(UserInputtedDrinkQuantity)

    If (error == true):

        output(DrinkQuantityError) #If user inputted drink quantity is not an integer, outputs an error message

    End If

    return error

End Function

Module orderValidation(drinkType, drinkSize, drinkQuantity):

    if (drinkType == ""): #checks that a drink type has been selected

        output(DrinkTypeError) #outputs an error message if a drink type has not been selected

        return true; #returns true if an error is present

    Else If: (drinkSize == ""):

        output(DrinkSizeError) #outputs an error message if a drink size has not been selected

        return true; #returns true if an error is present

    Else If: ((drinkQuantityValidation(drinkQuantity)) == true):

        return true #returns true if an error is present

    Else:

        return false; #returns false if an error is not present

    End If

End Function

Module addItemToOrder(): #ButtonAddToOrder

    if ((orderValidation(drinkType, drinkSize, drinkQuantity)) == false):

        drinkPrice = calcDrinkPrice(drinkType,drinkSize)

        itemPrice = drinkQuantity \* drinkPrice

        const orderItem = [drinkType, drinkSize, drinkQuantity, drinkPrice, itemPrice]; #creates new instance of drinkOrder class using current drink configuration

        orderList.push(orderItem); #adds the new order item to the order list

        orderPrice = 0;

        calcOrderSubtotal()

        resetItem()

    End If

End Module

Module calcOrderSubtotal():

    orderSubtotal = 0;

    For (i in orderList)):

        orderPrice = orderList[i][4]

        orderSubtotal += orderPrice;

        updateOutputOrderSubtotal()

    End For Loop

    updateOrderTotal()

End

Module updateOutputOrderSubtotal():

    output(orderSubtotal)

End Module

Module updateOutputOrderDiscount():

    output(orderDiscount)

End Module

Module updateOutputOrderTotal():

    output(orderTotal)

End Module

Module updateOrderTotal():

    If (primaBeansMember == true):

        orderDiscount = orderSubtotal \* 0.1

    Else:

        orderDiscount = 0

    End If

    orderTotal = orderSubtotal - orderDiscount

    updateOutputOrderDiscount()

    updateOutputOrderTotal()

End

Module completeOrder():

    add primaBeansMember to start of orderList

    append orderList to reportList

    resetOrder()

    updateAllOrderOutputs()

End

Module updateAllOrderOutputs():

    output(orderSubtotal)

    output(orderDiscount)

    output(orderTotal)

End

Module generateReport():

    resetReportTotals()

    For (order in reportList):

        If (reportList[order][0] == true): #checks if order is a Prima Beans Member

            For (drink in (reportList-1)): #if order is a Prima Beans Member, calculates total with discount

                reportDrinkType = reportList[order][(drink+1)][0]

                reportItemPrice = reportList[order][(drink+1)][5] \* 0.1

                calcReportTotals(reportDrinkType,reportItemPrice)

            End For Loop

        Else:

            For (drink in reportList[order]): #if order is not a Prima Beans Member, calculates total without discount

                reportDrinkType = reportList[order][(drink+1)][0]

                reportItemPrice = reportList[order][(drink+1)][5]

                calcReportTotals(reportDrinkType,reportItemPrice)

            End For Loop

        End If

    End For Loop

    updateReportTotals()

End Module

Module calcReportTotals(drinkType,itemPrice){

    if (drinkType == "Espresso Shot"):

        totalEspressoShot += itemPrice

    Else If: (drinkType == "Long Mac"):

        totalLongMac += itemPrice

    Else If: (drinkType == "Cappucino"):

        totalCappucino += itemPrice

    Else If: (drinkType == "Flat White"):

        totalFlatWhite += itemPrice

    End If

    reportTotal += itemPrice

End Module

Module updateReportTotals(): #populates report total table

    output(totalFlatWhite)

    output(totalCappucino)

    output(totalLongMac)

    output(totalEspressoShot)

    output(reportTotal)

End Module

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ RESETS ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

function resetOrder() {

    drinkPrice = 0;

    drinkType = "";

    drinkSize = "";

    drinkQuantity = 0;

    UserInputtedDrinkQuantity = 0;

    orderSubtotal = 0;

    orderTotal = 0;

    orderList = [];

    primaBeansMember = false;

    orderDiscount = 0;

    updateAllOrderOutputs()

}

function resetItem() {

    drinkPrice = 0;

    drinkType = "";

    drinkSize = "";

    drinkQuantity = 0;

    UserInputtedDrinkQuantity = 0;

    primaBeansMember = false;

    updateAllOrderOutputs()

}

function resetAll() {

    drinkPrice = 0;

    drinkType = "";

    drinkSize = "";

    drinkQuantity = 0;

    UserInputtedDrinkQuantity = 0;

    orderSubtotal = 0;

    orderTotal = 0;

    orderList = [];

    reportList = [];

    primaBeansMember = false;

    orderDiscount = 0;

    updateReportTotals()

    updateAllOrderOutputs()

}

function resetReportTotals() {

    totalEspressoShot = 0;

    totalLongMac = 0;

    totalCappucino = 0;

    totalFlatWhite = 0;

    reportTotal = 0;

    reportDrinkType = 0;

    reportItemPrice = 0;

}

Code:

HTML:

<!DOCTYPE html>

<html>

<head>

  <link rel="stylesheet" href="styles.css">

  <script type="text/javascript" src="script.js"></script>

</head>

<body>

<!-- Report Interface -->

<div id="reportInterface">

  <!-- Navigation Sizebar -->

  <div id="buttonNavigationSidebar">

    <button id="switchToOrderInterface" class="buttonSidebar" type="button" onclick="switchToOrderInterface()"> Order </button>

    <button id="switchToReportInterface" class="buttonSidebar" type="button" onclick="switchToReportInterface()"> Report </button>

    <button id="resetAll" class="buttonSidebar" type="button" onclick="resetAll()"> Reset </button>

    <button id="exit" class="buttonSidebar" type="button" onclick="exit()"> Exit </button>

  </div>

  <div id="panelReport">

    <h2>Drink Type</h2>

    <h2>Total</h2>

    <p>Flat White</p>

    <p id="totalFlatWhite"></p>

    <p>Cappucino</p>

    <p id="totalCappucino"></p>

    <p>Long Mac</p>

    <p id="totalLongMac"></p>

    <p>Espresso Shot</p>

    <p id="totalEspressoShot"></p>

    <h2>Total</h2>

    <h2 id="reportTotal"></h2>

  </div>

</div>

<!-- Order Interface -->

<div id="orderInterface">

  <!-- Navigation Sizebar -->

  <div id="buttonNavigationSidebar">

    <button id="switchToOrderInterface" class="buttonSidebar" type="button" onclick="switchToOrderInterface()"> Order </button>

    <button id="switchToReportInterface" class="buttonSidebar" type="button" onclick="switchToReportInterface()"> Report </button>

    <button id="resetAll" class="buttonSidebar" type="button" onclick="resetAll()"> Reset </button>

    <button id="exit" class="buttonSidebar" type="button" onclick="exit()"> Exit </button>

  </div>

  <!-- Drink Type -->

  <div id="panelDrinkType">

    <button id="flatWhite" type="button" onclick="flatWhite()"> flatWhite </button>

    <button id="cuppacino" type="button" onclick="cappucino()"> cappucino </button>

    <button id="longMac" type="button" onclick="longMac()"> longMac </button>

    <button id="espressoShot" type="button" onclick="espressoShot()"> espressoShot </button>

  </div>

  <!-- Drink Size -->

  <div id="panelDrinkSize">

    <button id="drinkSizeSmall" type="button" onclick="drinkSizeSmall()"> drinkSizeSmall </button>

    <button id="drinkSizeLarge" type="button" onclick="drinkSizeLarge()"> drinkSizeLarge </button>

  </div>

  <!-- Drink Quantity -->

  <div id="panelDrinkQuantity">

    <div id="wrapperInputAndButtonDrinkQuantity">

      <button type="button" onclick="decreaseDrinkQuantity()"> - </button>

      <input id="inputDrinkQuantity" type="text" onchange="updateDrinkQuantity()"></input>

      <button type="button" onclick="increaseDrinkQuantity()"> + </button>

    </div>

    <div id="wrapperAddToOrder">

      <button id="addItemToOrder" type="button" onclick="addItemToOrder()"> Add Drink to Order </button>

    </div>

    <div id="wrapperErrorMessage">

      <p id="outputDrinkQuantityError" class="errorMessage">Please enter a drink quantity, in the form of a positive integer greater than zero.</p>

      <p id="outputDrinkTypeError" class="errorMessage">Please select a drink type.</p>

      <p id="outputDrinkSizeError" class="errorMessage">Please select a drink size.</p>

    </div>

  </div>

  <!-- Order Summary -->

  <div id="panelOrderSummary">

    <div id="wrapperPrimaBeansYN">

      <p style="margin: 5px">Prima Beans Member:</p>

      <button type="button" onclick="primaBeansYes()"> Y </button>

      <button type="button" onclick="primaBeansNo()"> N </button>

    </div>

    <div id="wrapperOrderSummary">

      <p id="outputDrinkPrice"></p>

      <p id="outputDrinkQuantity"></p>

  </div>

    <div id="wrapperOrderTotalSummary">

      <p id="outputOrderSubtotal">Subtotal: </p>

      <p id="outputOrderDiscount">Discount: </p>

      <p id="outputOrderTotal">Total: </p>

      <button type="button" onclick="completeOrder()"> Complete Order </button>

    </div>

  </div>

</div>

</body>

</html>

CSS:

#reportInterface {

    display: none;

    grid-template-columns: 5% 95%;

    grid-gap: 10px;

    height: 100vh;

}

#panelReport {

    display: grid;

    grid-template-columns: 50% 50%;

    padding: 5%;

    align-items: center;

}

#orderInterface {

    display: grid;

    grid-template-columns: 5% 20% 20% 20% 30%;

    grid-gap: 10px;

    height: 100vh;

}

#panelDrinkType {

    display: grid;

    grid-template-rows: 25% 25% 25% 25%;

    grid-gap: 10px;

    height: 100vh;

}

#panelDrinkSize {

    display: grid;

    grid-template-rows: 25% 25%;

    grid-gap: 10px;

    height: 100vh;

}

#panelDrinkQuantity {

    display: grid;

    grid-template-rows: 25px 30px;

    grid-gap: 10px;

    height: 100vh;

}

#panelOrderSummary {

    display: grid;

    grid-template-rows: 1fr 1fr 1fr;

    height: 100vh;

    padding-inline: 10px;

}

#wrapperInputAndButtonDrinkQuantity {

    display: grid;

    grid-template-columns: 25px 50% 25px;

    justify-content: center;

}

#wrapperAddToOrder {

    display: flex;

    justify-content: center;

}

#wrapperPrimaBeansYN {

    display: grid;

    grid-template-columns: auto 25px 25px;

    height: 10px;

}

#inputDrinkQuantity {

    text-align: center;

}

.errorMessage {

    display: none;

}

.buttonSidebar {

    width: 100%;

    padding-top: 35%;

    padding-bottom: 35%;

    text-align: center;

    margin-bottom: 10px;

}

JS:

var drinkPrice = 0;

var drinkType = "";

var drinkSize = "";

var drinkQuantity = 0;

var UserInputtedDrinkQuantity = 0;

var orderSubtotal = 0;

var orderTotal = 0;

let orderList = [];

let reportList = [];

var primaBeansMember = false;

var orderDiscount = 0;

// Totals //

var totalEspressoShot = 0;

var totalLongMac = 0;

var totalCappucino = 0;

var totalFlatWhite = 0;

var reportTotal = 0;

var reportDrinkType = 0;

var reportItemPrice = 0;

// Interfaces //

function switchToOrderInterface() {

    document.getElementById("orderInterface").style.display = "grid"; //Unhides the order interface

    document.getElementById("reportInterface").style.display = "none"; //Hides the report interface

    resetOrder()

}

function switchToReportInterface() {

    document.getElementById("orderInterface").style.display = "none"; //Unhides the order interface

    document.getElementById("reportInterface").style.display = "grid"; //Hides the report interface

    resetOrder()

    generateReport()

}

// Drink Type //

function flatWhite() { //ButtonFlatWhite

    drinkType = "Flat White"; //Sets drinkType to "Flat White"

    showElementById("drinkSizeLarge"); //Unhides the ButtonDrinkSizeLarge

}

function cappucino() { //ButtonCappucino

    drinkType = "Cappucino"; //Sets drinkType to "Cappucino"

    showElementById("drinkSizeLarge"); //Unhides the ButtonDrinkSizeLarge

}

function longMac() { //ButtonLongMac

    drinkType = "Long Mac"; //Sets drinkType to "Long Mac"

    showElementById("drinkSizeLarge"); //Unhides the ButtonDrinkSizeLarge

}

function espressoShot() { //ButtonEspressoShot

    drinkType = "Espresso Shot"; //Sets drinkType to "Espresso Shot"

    drinkSize = "Small"

    hideElementById("drinkSizeLarge"); //Hides the ButtonDrinkSizeLarge as espresso shots do not come in a large size

}

function hideElementById(element) {

    document.getElementById(element).style.display = "none"; //Hides the element

}

function showElementById(element) {

    document.getElementById(element).style.display = "block"; //Unhides the element

}

function hideElementByClass(element) {

    classList = document.getElementsByClassName(element);

    for (i = 0; i < classList.length; i++) {

        classList[i].style.display = "none";

    }

}

// Drink Size //

function drinkSizeSmall() { //ButtonDrinkSizeSmall

    drinkSize = "Small"; //Sets drinkSize to "Small"

}

function drinkSizeLarge() { //ButtonDrinkSizeLarge

    drinkSize = "Large"; //Sets drinkSize to "Large"

}

// Drink Quantity //

function drinkQuantityIntegerValidation(UserInputtedDrinkQuantity) {

    if ((UserInputtedDrinkQuantity % 1) == 0) { //tests if UserInputtedDrinkQuantity is an integer

        return false; //returns false if there is no error, i,e. the UserInputtedDrinkQuantity is not an integer

    }

    else {

        return true; //returns true if there is an error, i.e. the UserInputtedDrinkQuantity is an integer

    }

}

function drinkQuantityLengthValidation(UserInputtedDrinkQuantity) {

    if ((UserInputtedDrinkQuantity.length) == 0) { //tests if UserInputtedDrinkQuantity is an integer

        return true; //returns true if there is an error, i,e. the UserInputtedDrinkQuantity is not an integer

    }

    else {

        return false; //returns false if there is no error, i.e. the UserInputtedDrinkQuantity is an integer

    }

}

function drinkQuantityParityValidation(UserInputtedDrinkQuantity) {

    if ((UserInputtedDrinkQuantity) < 0) { //tests if UserInputtedDrinkQuantity is an integer

        return true; //returns true if there is an error, i,e. the UserInputtedDrinkQuantity is not an integer

    }

    else {

        return false; //returns false if there is no error, i.e. the UserInputtedDrinkQuantity is an integer

    }

}

function updateDrinkQuantity() { //InputBoxDrinkQuantity

    UserInputtedDrinkQuantity = document.getElementById("inputDrinkQuantity").value; //Stores user inputted drink quantity

    if ((drinkQuantityLengthValidation(UserInputtedDrinkQuantity)) == false) {

        if ((drinkQuantityIntegerValidation(UserInputtedDrinkQuantity)) == false) {

            if ((drinkQuantityParityValidation(UserInputtedDrinkQuantity)) == false) {

                drinkQuantity = parseInt(UserInputtedDrinkQuantity);

                updateInputBoxDrinkQuantity();

                hideElementByClass("errorMessage"); //If user inputted drink quantity is not an integer, outputs an error message

            }

            else {

                drinkQuantity = 0;

                showElementById("outputDrinkQuantityError"); //If user inputted drink quantity is not an integer, outputs an error message

            }

        }

        else {

            drinkQuantity = 0;

            showElementById("outputDrinkQuantityError"); //If user inputted drink quantity is not an integer, outputs an error message

        }

    }

    else {

        drinkQuantity = 0;

        updateInputBoxDrinkQuantity()

        hideElementById("outputDrinkQuantityError"); //If user inputted drink quantity is not an integer, outputs an error message

    }

}

function updateInputBoxDrinkQuantity() { //InputBoxDrinkQuantity

    document.getElementById("inputDrinkQuantity").value = drinkQuantity; //displays the drinkQuantity in the input box

}

function decreaseDrinkQuantity() { //ButtonDrinkQuantityDecrease

    if (drinkQuantity > 0) { //Prevents drinkQuantity from being negative

        drinkQuantity -= 1; //Decreases drinkQuantity by 1

        updateInputBoxDrinkQuantity();

    }

}

function increaseDrinkQuantity() { //ButtonDrinkQuantityIncrease

    drinkQuantity += 1; //Increases drinkQuantity by 1

    updateInputBoxDrinkQuantity();

}

// Drink Price //

function calcDrinkPrice(drinkType, drinkSize) {

    if (drinkType == "Espresso Shot") {

        return 3;

    }

    else if (drinkType == "Long Mac") {

        if (drinkSize == "Small") {

            return 4.6;

        }

        else if (drinkSize == "Large") {

            return 5;

        }

        else {

            return "";

        }

    }

    else if (drinkType == "Cappucino") {

        if (drinkSize == "Small") {

            return 4.5;

        }

        else if (drinkSize == "Large") {

            return 5;

        }

        else {

            return "";

        }

    }

    else if (drinkType == "Flat White") {

        if (drinkSize == "Small") {

            return 4.3;

        }

        else if (drinkSize == "Large") {

            return 4.8;

        }

        else {

            return "";

        }

    }

    else {

        return "";

    }

}

// Prima Beans Member //

function primaBeansYes() {

    primaBeansMember = true

    updateOrderTotal()

}

function primaBeansNo() {

    primaBeansMember = false

    updateOrderTotal()

}

// Order //

function drinkQuantityValidation(UserInputtedDrinkQuantity) {

    error = drinkQuantityLengthValidation(UserInputtedDrinkQuantity)

    error = drinkQuantityIntegerValidation(UserInputtedDrinkQuantity)

    error = drinkQuantityParityValidation(UserInputtedDrinkQuantity)

    if (error == true) {

        showElementById("outputDrinkQuantityError"); //If user inputted drink quantity is not an integer, outputs an error message

    }

    else {

        hideElementById("outputDrinkQuantityError"); //If user inputted drink quantity is not an integer, outputs an error message

    }

    return error

}

function orderValidation(drinkType, drinkSize, drinkQuantity) {

    if (drinkType == "") { //checks that a drink type has been selected

        showElementById("outputDrinkTypeError"); ////outputs an error message if a drink type has not been selected

        return true; //returns true if an error is present

    }

    else if (drinkSize == "") {

        showElementById("outputDrinkSizeError"); ////outputs an error message if a drink size has not been selected

        return true; //returns true if an error is present

    }

    else if ((drinkQuantityValidation(drinkQuantity)) == true) {

        return true //returns true if an error is present

    }

    else {

        hideElementByClass("errorMessage"); //If user inputted drink quantity is not an integer, outputs an error message

        return false; //returns false if an error is not present

    }

}

function addItemToOrder() { //ButtonAddToOrder

    if ((orderValidation(drinkType, drinkSize, drinkQuantity)) == false) {

        drinkPrice = calcDrinkPrice(drinkType,drinkSize)

        itemPrice = drinkQuantity \* drinkPrice

        const orderItem = [drinkType, drinkSize, drinkQuantity, drinkPrice, itemPrice]; //creates new instance of drinkOrder class using current drink configuration

        orderList.push(orderItem); //adds the new order item to the order list

        orderPrice = 0;

        calcOrderSubtotal() //calculate and refresh orderSubtotal, orderTotal

        resetItem() //reset drink variables

    }

}

function calcOrderSubtotal() {

    orderSubtotal = 0; //resets orderSubtotal

    for (i = 0; i < (orderList.length); i++) { //loops through each order item

        orderPrice = orderList[i][4]

        orderSubtotal += orderPrice; //adds the order price of each item to the order subtotal

        updateOutputOrderSubtotal() //updates the displayed order subtotal

    }

    updateOrderTotal()

}

function updateOutputOrderSubtotal() {

    document.getElementById("outputOrderSubtotal").innerHTML = "Subtotal: $" + orderSubtotal.toFixed(2);

}

function updateOutputOrderDiscount() {

    document.getElementById("outputOrderDiscount").innerHTML = "Discount: $" + orderDiscount.toFixed(2);

}

function updateOutputOrderTotal() {

    document.getElementById("outputOrderTotal").innerHTML = "Total: $" + orderTotal.toFixed(2);

}

// Order Total //

function updateOrderTotal() {

    if (primaBeansMember == true) {

        orderDiscount = orderSubtotal \* 0.1

    }

    else {

        orderDiscount = 0;

    }

    orderTotal = orderSubtotal - orderDiscount

    updateOutputOrderDiscount()

    updateOutputOrderTotal()

}

function completeOrder() {

    orderList.unshift(primaBeansMember)

    reportList.push(orderList)

    resetOrder()

    updateAllOrderOutputs()

}

function updateAllOrderOutputs() {

    updateInputBoxDrinkQuantity()

    updateOutputOrderSubtotal()

    updateOutputOrderDiscount()

    updateOutputOrderTotal()

}

// Report //

function generateReport() {

    resetReportTotals()

    for (order = 0; order < (reportList.length); order++) {

        if (reportList[order][0] == true) { //checks if order is a Prima Beans Member

            for (drink = 1; drink < (reportList[order].length); drink++) {

                reportDrinkType = reportList[order][drink][0]

                reportItemPrice = reportList[order][drink][4] \* 0.9

                calcReportTotals(reportDrinkType,reportItemPrice)

            }

        }

        else {

            for (drink = 1; drink < (reportList[order].length); drink++) {

                reportDrinkType = reportList[order][drink][0]

                reportItemPrice = reportList[order][drink][4]

                calcReportTotals(reportDrinkType,reportItemPrice)

            }

        }

    }

    updateReportTotals()

}

function calcReportTotals(drinkType,itemPrice){

    if (drinkType == "Espresso Shot") {

        totalEspressoShot += itemPrice

    }

    else if (drinkType == "Long Mac") {

        totalLongMac += itemPrice

    }

    else if (drinkType == "Cappucino") {

        totalCappucino += itemPrice

    }

    else if (drinkType == "Flat White") {

        totalFlatWhite += itemPrice

    }

    reportTotal += itemPrice

}

function updateReportTotals() { //populates report total table

    document.getElementById("totalFlatWhite").innerHTML = "$" + totalFlatWhite.toFixed(2); //populates report total table with the total sales of flat whites with a $ and to two decimal places

    document.getElementById("totalCappucino").innerHTML = "$" + totalCappucino.toFixed(2); //populates report total table with the total sales of cappucinos with a $ and to two decimal places

    document.getElementById("totalLongMac").innerHTML = "$" + totalLongMac.toFixed(2); //populates report total table with the total sales of long macs with a $ and to two decimal places

    document.getElementById("totalEspressoShot").innerHTML = "$" + totalEspressoShot.toFixed(2); //populates report total table with the total sales of espresso shots with a $ and to two decimal places

    document.getElementById("reportTotal").innerHTML = "$" + reportTotal.toFixed(2); //populates report total table with the total sales of espresso shots with a $ and to two decimal places

}

// Reset //

function resetOrder() {

    drinkPrice = 0;

    drinkType = "";

    drinkSize = "";

    drinkQuantity = 0;

    UserInputtedDrinkQuantity = 0;

    orderSubtotal = 0;

    orderTotal = 0;

    orderList = [];

    primaBeansMember = false;

    orderDiscount = 0;

    updateAllOrderOutputs()

}

function resetItem() {

    drinkPrice = 0;

    drinkType = "";

    drinkSize = "";

    drinkQuantity = 0;

    UserInputtedDrinkQuantity = 0;

    primaBeansMember = false;

    updateAllOrderOutputs()

}

function resetAll() {

    drinkPrice = 0;

    drinkType = "";

    drinkSize = "";

    drinkQuantity = 0;

    UserInputtedDrinkQuantity = 0;

    orderSubtotal = 0;

    orderTotal = 0;

    orderList = [];

    reportList = [];

    primaBeansMember = false;

    orderDiscount = 0;

    updateReportTotals()

    updateAllOrderOutputs()

}

function resetReportTotals() {

    totalEspressoShot = 0;

    totalLongMac = 0;

    totalCappucino = 0;

    totalFlatWhite = 0;

    reportTotal = 0;

    reportDrinkType = 0;

    reportItemPrice = 0;

}

// Exit //

function exit() {

    resetAll()

    document.getElementById("orderInterface").style.display = "none"; //Unhides the order interface

    document.getElementById("reportInterface").style.display = "none"; //Hides the report interface

  }