|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| basketController / swapToMainMenuScene  Intended action: Goes back to the Main Menu upon clicking it. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to swap back to the Main Menu from its current user interface. | Upon going into the basket scene, click the Main Menu button.  (Doing actions here does not affect it as they are independent events) | The Scene goes back to the Main Menu | The Scene goes back to the Main Menu |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| basketController / finilizeOrder  Intended action: Finalizes an order to be added as the Store Orders to be exported. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program should be able to know that it is empty and there are no orders to finalize. | Go to the basket scene and immediately press the finalize orders button. | Text area update: Can’t place order with no items in basket! | Text area update: Can’t place order with no items in basket! |
| 2 | The program should be able to process orders without problems. | Order a small coffee with 0 addins. Go to the basket scene and finalize the order. | Text area update: Order Placed! | Text area update: Order Placed! |
| 3 | The program should empty the basket after finalizing it and know it is empty again. | Order a small coffee with 0 addins. Go to the basket scene and finalize the order. Attempt to press the button again. | Text area update: Can’t place order with no items in basket! | Text area update: Can’t place order with no items in basket! |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| basketController / addToListView  Intended action: Adds all orders into the list, which can be removed at will. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program should recognize an empty list and have nothing in it. | Go straight to the basket scene and look at the list to the right. | The list appears empty. | The list appears empty. |
| 2 | The program should recognize that there are orders. | Order a small coffee and go to the basket scene. | The list shows the coffee item there. | The list shows the coffee item there. |
| 3 | The program should be able to delete unwanted orders. | Order a small coffee and go to the basket scene. Click on the coffee item and select delete. | The coffee order disappears from the list. | The coffee order disappears from the list. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| donutsSceneController / swapToMainMenuScene  Intended action: Goes back to the Main Menu Scene. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to swap back to the Main Menu from its current user interface. | Upon going into the donuts scene, click the Main Menu button.  (Doing actions here does not affect it as they are independent events) | The Scene goes back to the Main Menu | The Scene goes back to the Main Menu |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| donutsSceneController / confirmOrder  Intended action: Confirms an order for a specified donut. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must have a donut type selected to process an order. | Go into the donut scene. Press confirm order. | Test area update: Must select Donut Type | Text area update: Must select Donut Type |
| 2 | The program must have a donut flavor selected to process an order. | Go to the donut scene. Select any donut type. Enter a value for quantity (optional), but leave the flavor unselected. Press confirm order. | Text area update: Must select Donut Flavor | Text area update: Must select Donut Flavor |
| 3 | The program must have a valid amount of donuts to process an order. | Go to the donut scene. Select any donut type and flavor. Enter a value less than or equal to 0 for amount. Press confirm order. | Text area update: Amount cannot be less than 0 | Text area update: Amount cannot be less than 0 |
| 4 | The program must have a valid amount of donuts to process an order. | Go to the donut scene. Select any donut type and flavor. Enter a non-numeric value for amount. Press confirm order. | Text area update: Amount must be a valid number! | Text area update: Amount must be a valid number! |
| 5 | The program must be able to properly process a yeast donut. | Go to the donut scene. Select a yeast donut, a flavor for it, and set the amount as 1 or any value greater than 0. Press confirm order. | Text area update: Order added to cart. | Text area update: Order added to cart. |
| 6 | The program must be able to process a cake donut. | Go to the donut scene. Select a cake donut, a flavor for it, and set the amount as 1 or any value greater than 0. Press confirm order. | Text area update: Order added to cart. | Text area update: Order added to cart. |
| 7 | The program must be able to process a donut hole. | Go to the donut scene. Select a donut hole, a flavor for it, and set the amount as 1 or any value greater than 0. Press confirm order. | Text area update: Order added to cart. | Text area update: Order added to cart. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| donutsSceneController / addToListView  Intended action: Updates the list of current donut orders and can be edited or deleted here. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to recognize an empty list. | Go straight to the donut scene and look at the right. | An empty list is seen with no orders. | An empty list is seen with no orders. |
| 2 | The program must be able to update the list when an order is processed. | Complete test case 5, 6, or 7 in confirmOrder, then look to the list on the right. | The order is there. | The order is there. |
| 3 | The program must be able to edit an order here. | Complete Test Case 2 above. Click on the order and select edit. | The order is removed, but all the fields have been updated to match the order. | The order is removed, but all the fields have been updated to match the order. |
| 4 | The program must be able to remove an order here. | Complete Test Case 2 above. Click on the order and select remove. | The order is removed from the list. | The order is removed from the list. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| storeOrderController / swapToMainMenuScene  Intended action: Goes back to the Main Menu Scene. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to swap back to the Main Menu from its current user interface. | Upon going into the store order scene, click the Main Menu button.  (Doing actions here does not affect it as they are independent events) | The Scene goes back to the Main Menu | The Scene goes back to the Main Menu |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| storeOrderController / exportData  Intended action: Exports the orders finalized. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must recognize its list is empty to export orders. | Go to the store order scene and immediately click the export button. | Text field update: No orders to export! | Text field update: No orders to export! |
| 2 | The program must be able to process all finalized orders and properly export them. | Order a small coffee with 0 addins. Finalize the order, then come to the store order scene and press export. | Text field update: All orders exported!  A new file is made as well: All Store Orders.txt | Text field update: All orders exported!  A new file is made as well: All Store Orders.txt |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| storeOrderController / addToListView  Intended action: Updates the list of all finalized orders here and able to delete them. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to recognize an empty list. | Go straight to the store order scene and look at the right. | An empty list is seen with no orders. | An empty list is seen with no orders. |
| 2 | The program must be able to update the list when an order is processed. | Order a small coffee with 0 addins, finalize this order, then go to the order scene and look to the right. | The order is there. | The order is there. |
| 3 | The program must be able to edit an order here. | Complete Test Case 2 above. Click on the order and press remove. | The order is removed from the list. | The order is removed from the list. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MainMenuController / swapToDonutScene  Intended action: Goes to the Donut Scene. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to open the scene on the main menu. | Click on the Order Donuts button. | Goes to the donut scene on a new window. | Goes to the donut scene on a new window. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MainMenuController / swapToStoreOrderScene  Intended action: Goes to the Store Order Scene | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to open the scene on the main menu. | Click on the Store Orders button. | The main menu changes to the store orders scene. | The main menu changes to the store orders scene. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MainMenuController / swapToCoffeeScene  Intended action: Goes to the Coffee Scene | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to open the scene on the main menu. | Click on the Order Coffee button. | The main menu changes to the coffee scene. | The main menu changes to the coffee scene. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MainMenuController / swapToOrderScene  Intended action: Goes to the Order Scene where orders are exported. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to open the scene on the main menu. | Click on the Cart button. | The main menu goes to the basket scene. | The main menu goes to the basket scene. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| coffeeController / swapToMainMenuScene  Intended action: Goes back to the Main Menu scene. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to swap back to the Main Menu from its current user interface. | Upon going into the coffee scene, click the Main Menu button.  (Doing actions here does not affect it as they are independent events) | The Scene goes back to the Main Menu | The Scene goes back to the Main Menu |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| coffeeController / numbersOnly  Intended action: Ensures that only numbers are entered into the quantity line. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program ensures only numbers can be entered into the quantity line. | Go to the coffee scene. On the quantity line, attempt to enter a non-numeric value (such as letters or symbols). | The typed value is immediately deleted. | The typed value is immediately deleted. |
| 2 | The program ensures only numbers can be entered into the quantity line. | Go to the coffee scene. On the quantity line, enter any numeric value. | The numeric value is entered and it stays. | The numeric value is entered and it stays. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| coffeeController / actionDone  Intended action: Processes the coffee order. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must have a coffee size in order to process an order. | Go to the coffee scene and immediately press submit order. | Text area update: Must select size first! | Text area update: Must select size first! |
| 2 | The program can assume a blank quantity line means only a singular coffee is being ordered. | Select a small coffee. (Addins optional). On the quantity line, leave it blank and press submit. | Text area update: Order added to cart.  A quantity of 1 is listed for the order. | Text area update: Order added to cart.  A quantity of 1 is listed for the order. |
| 3 | The program can handle more than one coffee quantity when entered. | Select a small coffee. (Addins optional). On the quantity line, enter a value greater than 1 and press submit. | Text area update: Order added to cart.  A quantity greater than 1 is listed for the order and the price reflects it. | Text area update: Order added to cart.  A quantity greater than 1 is listed for the order and the price reflects it. |
| 4 | The program can handle a plain small coffee with no errors. | Select a small coffee. Leave the quantity blank and have 0 addins. Press submit. | Text area update: Order added to cart. | Text area update: Order added to cart. |
| 5 | The program can handle a plain tall coffee with no errors. | Select a tall coffee. Leave the quantity blank and have 0 addins. Press submit. | Text area update: Order added to cart. | Text area update: Order added to cart. |
| 6 | The program can handle a plain grande coffee with no errors. | Select a grande coffee. Leave the quantity blank and have 0 addins. Press submit. | Text area update: Order added to cart. | Text area update: Order added to cart. |
| 7 | The program can handle a plain venti coffee with no errors. | Select a venti coffee. Leave the quantity blank and have 0 addins. Press submit. | Text area update: Order added to cart. | Text area update: Order added to cart. |
| 8 | The program can handle 1 addin for any coffee and update the price properly. | On the coffee scene, select a small coffee. Select any 1 addin (only 1). Press submit order. | Text area update: Order added to cart.  The price updated by $0.30 for 1 addin. | Text area update: Order added to cart.  The price updated by $0.30 for 1 addin. |
| 9 | The program can handle 2 addins for any coffee and update the price properly. | On the coffee scene, select a small coffee. Select any 2 addin (only 2). Press submit order. | Text area update: Order added to cart.  The price updated by $0.60 for 2 addins. | Text area update: Order added to cart.  The price updated by $0.60 for 2 addins. |
| 10 | The program can handle 3 addins for any coffee and update the price properly. | On the coffee scene, select a small coffee. Select any 3 addin (only 3). Press submit order. | Text area update: Order added to cart.  The price updated by $0.90 for 3 addins. | Text area update: Order added to cart.  The price updated by $0.90 for 3 addins. |
| 11 | The program can handle 4 addins for any coffee and update the price properly. | On the coffee scene, select a small coffee. Select any 4 addin (only 4). Press submit order. | Text area update: Order added to cart.  The price updated by $1.20 for 4 addins. | Text area update: Order added to cart.  The price updated by $1.20 for 4 addins. |
| 12 | The program can handle all addins for any coffee and update the price properly. | On the coffee scene, select a small coffee. Select all addins. Press submit order. | Text area update: Order added to cart.  The price updated by $1.50 for 5 addin. | Text area update: Order added to cart.  The price updated by $1.50 for 5 addin. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| coffeeController / addToListView  Intended action: Updates the list of all coffee orders completed and can edit or delete them. | | | | |
| Test Case # | Requirements | Test Description and Input Data | Expected Result / Output | Actual Result / Output |
| 1 | The program must be able to recognize an empty list. | Go straight to the coffee scene and look at the right. | An empty list is seen with no orders. | An empty list is seen with no orders. |
| 2 | The program must be able to update the list when an order is processed. | Go to the coffee scene and order a small coffee with 0 addins. Press submit order and look to the right. | The order is there. | The order is there. |
| 3 | The program must be able to edit an order here. | Complete Test Case 2 above. Click on the order and select edit. | The order is removed, but all the fields have been updated to match the order. | The order is removed, but all the fields have been updated to match the order. |
| 4 | The program must be able to remove an order here. | Complete Test Case 2 above. Click on the order and select remove. | The order is removed from the list. | The order is removed from the list. |