**MCO1 TEST CASES**

**Regular Vending Machine Class**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *purchaseItem* | 1 | Determines if item can be bought when quantity is 15 | index: 1, quantity: 5 | 10 | 10 | P |
|  | 2 | Determines if item can be bought when quantity is 0 | index: 2, quantity: 11 | Print: Item Amount Exceeded. | Print: Item Amount Exceeded. | P |
|  | 3 | Determines if item can be bought when quantity is not enough. Initial Quantity: 3 | index: 3, quantity: 4 | Print: Item Amount Exceeded. | Print: Item Amount Exceeded. | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *findItem* | 1 | Determines if item already exists within Item Slots | Fried Egg | TRUE | TRUE | P |
|  | 2 | Determines if item does not exist within Item Slots | Coke | FALSE | FALSE | P |
|  | 3 | Determines if non-existing item is within Item Slots | Sprite | FALSE | FALSE | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *insertPayment* | 1 | Determines if amount inserted is a valid denomination | 200 | TRUE | TRUE | P |
|  | 2 | Determines if float value is a valid denomination | 13.5 | FALSE | FALSE | P |
|  | 3 | Determines if a negative integer is a valid denomination | -50 | FALSE | FALSE | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *calculateChange* | 1 | Determines change given based on payment and total price | pay: 100, price 75 | Return: 25 | 25 | P |
|  | Print denominations needed: 200x0, 100x0, 50x0, 20x1, 10x0, 5x1, 1x0 | Print denominations needed: 200x0, 100x0, 50x0, 20x1, 10x0, 5x1, 1x0 |
|  | 2 | Determines change given based on payment and total price | pay: 200, price 100 | Return: 100 | Return: 100 | P |
|  | Print denominations needed: 200x0, 100x1, 50x0, 20x0, 10x0, 5x0, 1x0 | Print denominations needed: 200x0, 100x1, 50x0, 20x0, 10x0, 5x0, 1x0 |
|  | 3 | Determines change given based on payment and total price | pay: 250, price 75 | Return: 175 | Return: 175 | P |
|  | Print denominations needed: 200x0, 100x1, 50x1, 20x1, 10x0, 5x1, 1x0 | Print denominations needed: 200x0, 100x1, 50x1, 20x1, 10x0, 5x1, 1x0 |

**Item Slot Class**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *getItemAvailability* | 1 | Determines if item is available when quantity is 0 | 0 | FALSE | FALSE | P |
|  | 2 | Determines if item is available when quantity is 10 | 10 | TRUE | TRUE | P |
|  | 3 | Determines if item is available when quantity is 15 | 15 | TRUE | TRUE | P |

**Item Class**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *isAllowSell* | 1 | Determines if item can be sold when requested exceeds quantity (15) | 16 | FALSE | FALSE | P |
|  | 2 | Determines if item can be sold when requested is less than quantity (12) | 5 | TRUE | TRUE | P |
|  | 3 | Determines if item can be sold when requested is a negative value | -7 | FALSE | FALSE | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *canRestock* | 1 | Determines if item can be restocked when quantity is FULL (15) | 5 | FALSE | FALSE | P |
|  | 2 | Determines if item can be restocked when empty (0) | 15 | TRUE | TRUE | P |
|  | 3 | Determines if item can be restocked when quantity is 14 (MAX: 15) | 2 | FALSE | FALSE | P |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *restockItem* | 1 | Increments item quantity based on specified amount (Quantity: 14) | 1 | 15 | 15 | P |
|  | 2 | Increments item quantity based on specified amount (Quantity: 0) | 15 | 15 | 15 | P |
|  | 3 | Increments item quantity based on specified amount (Quantity: 7) | 7 | 14 | 14 | P |

**Money Box Class**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *isEmpty* | 1 | Determines if money within object is empty | 1000 | FALSE | FALSE | P |
|  | 2 | Determines if money within object is empty | 0 | TRUE | TRUE | P |
|  | 3 | Determines if money within object is empty | -100 | NOT POSSIBLE | FALSE | F |
|  |  |  |  |  |  |  |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *addMoney* | 1 | Increments money by given amount when Money is 1000 | 500 | 1500 | 1500 | P |
|  | 2 | Increments money by given amount when Money is 500 | 500 | 1000 | 1000 | P |
|  | 3 | Increments money by given amount when Money is 0 | 200 | 200 | 200 | P |
|  |  |  |  |  |  |  |
| ***Method*** | ***#*** | ***Test Description*** | ***Sample Input Data*** | ***Expected Output*** | ***Actual Output*** | ***P/F*** |
| *addEarnings* | 1 | Increments earnings by given amount when Earnings is 1000 | 500 | 1500 | 1500 | P |
|  | 2 | Increments earnings by given amount when Earnings is 500 | 500 | 1000 | 1000 | P |
|  | 3 | Increments earnings by given amount when Earnings is 0 | 200 | 200 | 200 | P |