Controller.java

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
AddMoneytoSystem	1	All text fields in BillInsert form is valid and filled	Money1: 30 Money5: 4 Money10: 2 Money20: 1 Money50: 1 Money100: 1 Money200: 1 Money500: 1 Money1000: 1	A banknotes object with the given user inputs is created and returned	A banknotes object with the given user inputs is created and returned	P
	2	Some text fields are filled, others are left blank	Money1: 1 Money5: Money10: 2 Money20: Money50: Money100: 1 Money200: Money500: Money1000:	A banknotes object with the given user inputs is created and returned, blank text fields have their values set to 0	A banknotes object with the given user inputs is created and returned, blank text fields have their values set to 0	P
	3	Negative values appear in text field	Money1: -1 Money5: 0 Money10: 2 Money20: 1 Money50: 1 Money100: 1 Money200: 1 Money500: 1 Money1000: 1	Invalid input error message pops up. A banknotes object with all values set to 0 is returned	Invalid input error message pops up. A banknotes object with all values set to 0 is returned	P

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
stockManage	1	ItemName belongs to an item in slots and addStock is a positive integer	ItemName: Potato addStock: 5	Potato Objects(Slots [0].size-1): 15	Potato Objects(Slots [0].size-1): 15	P
	2	ItemName belongs to an item in slots and addStock exceeds the item capacity	ItemName: Bacon addStock: 11	A popup appears telling the user the process was unsuccessful	A popup appears telling the user the process was unsuccessful	P
	3	ItemName does not belong to an item in slots	ItemName: Matcha addStock: 5	A popup appears telling the user the	A popup appears telling the user the	P

				process was unsuccessful	process was unsuccessful	

Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
priceManage	1	ItemName belongs to an item in slots and newPrice is a positive integer	ItemName: Gravy newPrice: 25	ItemPrice: 25	ItemPrice: 25	P
	2	ItemName belongs to an item in slots and newPrice is a negative integer	ItemName: Cheese newPrice: -25	A popup appears telling the user the input is invalid	A popup appears telling the user the input is invalid	P
	3	ItemName does not belong to an item in slots	ItemName: Grimace Shake newPrice: 25	A popup appears telling the user the input is invalid	A popup appears telling the user the input is invalid	P

Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
replenishManage	1	All inputs are positive integers	oneP: 0 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 0 oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 add1: 20 add5: 10 add10: 4 add20: 5 add50: 2 add100: 4 add200: 1 add500: 1 add1000: 1	oneP: 20 fiveP: 10 tenP: 4 twentyP: 5 fiftyP: 2 oneHunP: 4 twoHunP: 1 fiveHunP: 1 oneKP: 1 Total: 2410	oneP: 20 fiveP: 10 tenP: 4 twentyP: 5 fiftyP: 2 oneHunP: 4 twoHunP: 1 fiveHunP: 1 oneKP: 1 Total: 2410	P
	2	There is a negative integer present	add20: -5	A popup appears telling the user the input	A popup appears telling the user the input	P

				is invalid	is invalid	
	3	User inputs a value in just one text field	oneP: 20 add1: 20	oneP: 40	oneP: 40	P

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
collectionManage	1	All inputs are positive integers and amount is present in the machine	oneP: 40 fiveP: 10 tenP: 4 twentyP: 5 fiftyP: 2 oneHunP: 4 twoHunP: 1 fiveHunP: 1 oneKP: 1 Input1: 1 Input5: 1 Input10: 1 Input20: 1 Input50: 1 Input100: 1 Input200: 1 Input500: 1 Input1000: 1	oneP: 39 fiveP: 9 tenP:3 twentyP: 4 fiftyP: 1 oneHunP: 3 twoHunP: 0 fiveHunP: 0 oneKP: 0 Total: 544	oneP: 39 fiveP: 9 tenP:3 twentyP: 4 fiftyP: 1 oneHunP: 3 twoHunP: 0 fiveHunP: 0 oneKP: 0 Total: 544	P
	2	There is a negative integer present	Input20: -5	A popup appears telling the user the input is invalid	A popup appears telling the user the input is invalid	P
	3	User tries to take more money than what is present in the machine	Input1: 9	oneP: 30	oneP: 30	P

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
buyItem	1	Item slot input is not valid	nSlotChoice: 100	Error message containing “Invalid Item Slot Input!” pops up	Error message containing “Invalid Item Slot Input!” pops up	P

	2	Item slot input is valid but quantity input is a negative value	nSlotChoice: 2 nQtyChoice: -1	Error message containing “Invalid Quantity Input!” pops up	Error message containing “Invalid Quantity Input!” pops up	P
	3	Item slot and quantity inputs are both valid but machine doesn’t have enough stock	nSlotChoice: 2 nQtyChoice: 5 CurrentM.getSlots()[2].size = 4	Error message containing “Invalid Quantity Input!” pops up	Error message containing “Invalid Quantity Input!” pops up	P
	4	Item slot and quantity inputs are both valid but machine is out of stock	nSlotChoice: 2 nQtyChoice: 5 CurrentM.getSlots()[2].size = 1	Error message containing “Item Out of Stock!” pops up	Error message containing “Item Out of Stock!” pops up	P
	5	Item slot and quantity input are both valid and machine has enough stock	nSlotChoice: 2 nQtyChoice: 2 CurrentM.getSlots()[2].size = 11	Information on userMoney, nSlotChoice and nQtyChoice gets brought to CurrentM.BuyItem	Information on userMoney, nSlotChoice and nQtyChoice gets brought to CurrentM.BuyItem	P

Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
QtyChecker	1	Item is out of stock	Qty[0] = 2 CurrentM.getSlots()[0].size = 1	validQty = false Error message containing “Invalid Quantity Input!” shows up	validQty = false Error message containing “Invalid Quantity Input!” shows up	P
	2	Quantity wanted is greater than stock	Qty[0] = 5 CurrentM.getSlots()[0].size = 3	validQty = false Error message	validQty = false Error message	P

				containing “Invalid Quantity Input!” shows up	containing “Invalid Quantity Input!” shows up	
	3	Quantity wanted is a negative value	Qty[0] = -4	validQty = false Error message containing “Invalid Quantity Input!” shows up	validQty = false Error message containing “Invalid Quantity Input!” shows up	P
	4	Quantity wanted is valid and there is enough stock	Qty[0] = 5 CurrentM.getSlots()[0].size = 7	validQty = true	validQty = true	P

Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
buyPremade	1	User selects a premade meal slot that’s valid, the ingredients to make the meal is still in stock	specialSlot: 5 [mashed potato] CurrentM.getSlots()[0].size = 9 CurrentM.getSlots()[1].size = 9 CurrentM.getSlots()[2].size = 9	Information on the quantity and userMoney gets brought to BuyCustom in SpecialVM.java	Information on the quantity and userMoney gets brought to BuyCustom in SpecialVM.java	P
	2	User selects a premade meal slot that’s valid, but at least one of the ingredients to make the meal is out of stock	specialSlot: 5 [mashed potato] CurrentM.getSlots()[0].size = 9 CurrentM.getSlots()[1].size = 1 CurrentM.getSlots()[2].size = 9	Error message pops up indicating invalid quantity	Error message pops up indicating invalid quantity	P
	3	User selects a premade meal slot that	specialSlot: 50	Error message	Error message	P

		is invalid (out of bounds)		pops up indicating invalid input	pops up indicating invalid input	

Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
buyCustom	1	User inputs quantities that result in the creation of a special Product	Qty[0]: 1 Qty[1]: 1 Qty[2]: 1 Qty[5]: 1	Prompts the user the process string and the special product “Poutine” Objects: Slots[0].size-1: 14 Slots[1].size-1: 9 Slots[2].size-1: 9 Slots[5].size-1: 9 (Subtracts 1 for all affected)	Prompts the user the process string and the special product “Poutine” Objects: Slots[0].size-1 : 14 Slots[1].size-1 : 9 Slots[2].size-1 : 9 Slots[5].size-1 : 9 (Subtracts 1 for all affected)	P
	2	User buys an item individually	Qty[0]: 1	Slots[0].size-1: 9 (Subtracts by Qty input)	Slots[0].size-1 : 9 (Subtracts by Qty input)	P
	3	User buys an item individually which cannot be bought alone	Qty[5]: 1	Prompts the user that the item cannot be bought individually	Prompts the user that the item cannot be bought individually	P

VendingMachine.java

Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
restockItem	1	ItemName belongs to an item in slots and addStock is a positive	ItemName: Potato	restockedItem: 0	restockedItem : 0	P

		integer	addStock: 5			
	2	ItemName belongs to an item in slots and addStock exceeds the item capacity	ItemName: Bacon addStock: 11	restockedItem: -1	restockedItem: -1	P
	3	ItemName does not belong to an item in slots	ItemName: Matcha addStock: 5	restockedItem: -1	restockedItem: -1	P

Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
editPrice	1	ItemName belongs to an item in slots and newPrice is a positive integer	ItemName: Gravy newPrice: 25	modItem: 2	modItem: 2	P
	2	ItemName belongs to an item in slots and newPrice is a negative integer	ItemName: Cheese newPrice: -25	modItem: -1	modItem: -1	P
	3	ItemName does not belong to an item in slots	ItemName: Grimace Shake newPrice: 25	modItem: -1	modItem: -1	P

Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
takeMoneyfromSystem	1	All input amounts are collectable from the machine	money1: 1 money5: 1 money10: 1 money20: 1 money50: 1 money100: 1 money200: 1 money500: 1 money1000: 1 In the machine: oneP: 4 fiveP: 4 tenP: 4 twentyP: 4	In the machine: oneP: 3 fiveP: 3 tenP: 3 twentyP: 3 fiftyP: 3 oneHunP: 3 twoHunP: 3 fiveHunP: 3 oneKP: 0 Total: 2658 Collected amount:	In the machine: oneP: 3 fiveP: 3 tenP: 3 twentyP: 3 fiftyP: 3 oneHunP: 3 twoHunP: 3 fiveHunP: 3 oneKP: 0 Total: 2658 Collected amount:	P

			fiftyP: 4 oneHunP: 4 twoHunP: 4 fiveHunP: 4 oneKP: 1 Total: 4544	1886	1886	
	2	Some inputs exceed the amount in the machine	money1: 3 money5: 4 money10: 3 money20: 3 money50: 3 money100:3 money200: 4 money500: 4 money1000: 1 In the machine: oneP: 3 fiveP: 3 tenP: 3 twentyP: 3 fiftyP: 3 oneHunP: 3 twoHunP: 3 fiveHunP: 3 oneKP: 0 Total: 2658	Error message pops up for each invalid collection amount In the machine: oneP: 0 fiveP: 3 tenP: 0 twentyP: 0 fiftyP: 0 oneHunP: 0 twoHunP: 3 fiveHunP: 3 oneKP: 0 Total: 2115 Collected amount: 543	Error message pops up for each invalid collection amount In the machine: oneP: 0 fiveP: 3 tenP: 0 twentyP: 0 fiftyP: 0 oneHunP: 0 twoHunP: 3 fiveHunP: 3 oneKP: 0 Total: 2115 Collected amount: 543	P
	3	None of the inputs can be collected from the vending machine	money1: 4 money5: 4 money10: 4 money20: 4 money50: 4 money100:4 money200: 4 money500: 4 money1000: 4 In the machine: oneP: 0 fiveP: 3 tenP: 0 twentyP: 0 fiftyP: 0 oneHunP: 0 twoHunP: 3	Error message pops up for each invalid collection amount In the machine: oneP: 0 fiveP: 3 tenP: 0 twentyP: 0 fiftyP: 0 oneHunP: 0 twoHunP: 3 fiveHunP: 3 oneKP: 0	Error message pops up for each invalid collection amount In the machine: oneP: 0 fiveP: 3 tenP: 0 twentyP: 0 fiftyP: 0 oneHunP: 0 twoHunP: 3 fiveHunP: 3 oneKP: 0	P

			fiveHunP: 3 oneKP: 0 Total: 2115	Total: 2115 Collected amount: 0	Total: 2115 Collected amount: 0	

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
BuyItem	1	User has enough money to buy the item, machine has enough change	userMoney: oneP: 1 fiveP: 1 tenP: 1 twentyP: 1 fiftyP: 1 oneHunP: 1 twoHunP: 1 fiveHunP: 1 oneKP: 1 In the machine: oneP: 10 fiveP: 5 tenP: 5 twentyP: 5 fiftyP: 5 oneHunP: 5 twoHunP: 5 fiveHunP: 5 oneKP: 1 Slot input: 0 [potato, 20p] Quantity input: 1	userMoney: oneP: 1 fiveP: 1 tenP: 1 twentyP: 0 fiftyP: 1 oneHunP: 1 twoHunP: 1 fiveHunP: 1 oneKP: 1 In the machine: oneP: 10 fiveP: 5 tenP: 5 twentyP: 6 fiftyP: 5 oneHunP: 5 twoHunP: 5 fiveHunP: 5 oneKP: 1 “x1 Potato bought!” message pops up Potato quantity decreases	userMoney: oneP: 1 fiveP: 1 tenP: 1 twentyP: 0 fiftyP: 1 oneHunP: 1 twoHunP: 1 fiveHunP: 1 oneKP: 1 In the machine: oneP: 10 fiveP: 5 tenP: 5 twentyP: 6 fiftyP: 5 oneHunP: 5 twoHunP: 5 fiveHunP: 5 oneKP: 1 “x1 Potato bought!” message pops up Potato quantity decreases	P
	2	User does not have enough money to buy the item	userMoney: oneP: 1 fiveP: 0 tenP: 0 twentyP: 0	Error message containing “Not enough money inserted!	Error message containing “Not enough money inserted!	P

			fiftyP: 0 oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 Slot input: 0 [potato, 20p] Quantity input: 1	Transaction Cancelled" pops up Transaction doesn't happen	Transaction Cancelled" pops up Transaction doesn't happen	
	3	User has enough money to buy the item, but machine does not have enough change	userMoney: oneP: 0 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 1 oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 In the machine: oneP: 10 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 0 oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 Slot input: 0 [potato, 20p] Quantity input: 1	Error message containing "Error in producing change!" pops up Transaction doesn't happen	Error message containing "Error in producing change!" pops up Transaction doesn't happen	P

SpecialVM.java

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
BuyCustom	1	User doesn't have enough money to buy the custom product	userMoney: oneP: 10 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 0	Error message containing "Not enough money inserted!	Error message containing "Not enough money inserted!	P

			oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 nQtyChoice[0]: 1 nQtyChoice[5]: 1 [cost: 23]	Transaction Cancelled" pops up Transaction doesn't happen	Transaction Cancelled" pops up Transaction doesn't happen	
	2	User has enough money to buy the custom product but machine doesn't have enough money to produce change	userMoney: oneP: 0 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 1 oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 In the machine: oneP: 10 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 0 oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 nQtyChoice[0]: 1 nQtyChoice[5]: 1 [cost: 23]	Error message containing "Error in producing change!" pops up Transaction doesn't happen	Error message containing "Error in producing change!" pops up Transaction doesn't happen	P
	3	User has enough money to buy the custom product and machine has enough for change	userMoney: oneP: 0 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 1	userMoney: oneP: 27 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 0	userMoney: oneP: 27 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 0	P

			oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 In the machine: oneP: 30 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 0 oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 nQtyChoice[0]: 1 nQtyChoice[5]: 1 [cost: 23]	oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 In the machine: oneP: 3 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 1 oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 Message containing process and product name pops up Quantity for potatoes and salt also decrease	oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 In the machine: oneP: 3 fiveP: 0 tenP: 0 twentyP: 0 fiftyP: 1 oneHunP: 0 twoHunP: 0 fiveHunP: 0 oneKP: 0 Message containing process and product name pops up Quantity for potatoes and salt also decrease	

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
buildProduct	1	The inputs result in one of the special products	Qty[0]: 1 Qty[1]: 1 Qty[3]: 1 Qty[5]; 1	Product: Cheesy Bacon Fries	Product: Cheesy Bacon Fries	P
	2	The inputs result in a normal product	Qty[0]: 4 Qty[1]: 1 Qty[5]: 1	Product:Medium Cheese Fries	Product:Medium Cheese Fries	P
	3	The inputs does not contain potatoes	Qty[1]: 1 Qty[3]: 1 Qty[4]: 1 Qty[7]: 1	Product: Spicy Cheese BBQ Bacon	Product: Spicy Cheese BBQ Bacon	P

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
processProduct	1	The inputs result in one of the special products	Qty[0]: 1 Qty[1]: 1 Qty[3]: 1 Qty[5]; 1	Peeling and cutting potatoes... Processing Potatoes... Putting potatoes into container... Frying bacon... Putting bacon into container.. Melting cheese... Salting it up... The aroma of freshly cooked food wafts through the air! Enjoy your Cheesy Bacon Fries! Calories: (165g)	Peeling and cutting potatoes... Processing Potatoes... Putting potatoes into container... Frying bacon... Putting bacon into container.. Melting cheese... Salting it up... The aroma of freshly cooked food wafts through the air! Enjoy your Cheesy Bacon Fries! Calories: (165g)	P
	2	The inputs result in a normal product and theres multiples of one item	Qty[0]: 4 Qty[1]: 1 Qty[5]: 1	Peeling and cutting potatoes... Processing Potatoes... Putting potatoes into container... Melting cheese.. Salting it up... The aroma of freshly cooked food wafts through the air! Enjoy your Medium Cheese Fries! Calories: (420g)	Peeling and cutting potatoes... Processing Potatoes... Putting potatoes into container... Melting cheese.. Salting it up... The aroma of freshly cooked food wafts through the air! Enjoy your Medium Cheese Fries! Calories: (420g)	P
	3	The inputs does not contain potatoes	Qty[1]: 1 Qty[3]: 1 Qty[4]: 1	Frying bacon.. Putting bacon into container...	Frying bacon.. Putting bacon	P

			Qty[7]: 1	Slicing chili... Putting chili into container... Melting cheese... Adding barbeque powder... The aroma of freshly cooked food wafts through the air! Enjoy your Spicy Cheese BBQ Bacon! Calories: (80g)	into container... Slicing chili... Putting chili into container... Melting cheese... Adding barbeque powder... The aroma of freshly cooked food wafts through the air! Enjoy your Spicy Cheese BBQ Bacon! Calories: (80g)	