Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
CashRegister Class						
receivePaymentModel()	1	payment is less than the required amount	required amount: 100; payment: 50	FALSE	FALSE	P
	2	payment exceeds the required amount and there is enough change	required amount: 100; payment: 100	TRUE	TRUE	p
	3	payment exceeds the required amount but there is not enough change available	required amount: 100; payment: 200	FALSE	FALSE	p
giveMoneyModel()	1	Determines if the change is returned proper;y to the user properly	required amount: 100; payment: 200	denomination 100: 1	denomination 100: 1	p
	2	Determines if the payment is returned properly when the machine has no change	payment: 100; change in the machine: 0	denomination 100: 1	denomination 100: 1	p
	3	Determines if the payment is returned properly when it's incorrect	payment: 100; required amount: 200	denomination 100: 1	denomination 100: 1	p
isEnoughChange()	1	Determines if the change in the vending machine is enough when the payment is greater than amount to be paid	required amount: 100; payment: 200	TRUE	TRUE	p
	2	Determines if the change in the machine is not enough when the payment is greater than the amount to be paid	required amount: 200; payment: 100	TRUE	TRUE	p
	3	Determines if the change in the machine is enough when the payment is equal to the amount to be paid	required amount: 200; payment: 200	TRUE	TRUE	p
DecreseChange()	1	Determines if a money instance is removed	decrease PHP 5 (PHP 5 count = 5)	PHP 5 count = 4	PHP 5 count = 4	P
	2	Determines if more than 1 money instance is removed	decrease PHP 10 thrice (PHP 5 count = 5)	PHP 10 count = 2	PHP 10 count = 2	P
	3	Determines if a money instance is removed when the denomination is not valid	decrease 220 denomination count	throw illegalStateException	throw illegalStateException	p
replenishChange()	1	Determines if a money instance is added	creates a new instance of PHP 5 (PHP 5 count = 6)	PHP 5 count = 7	PHP 5 count = 7	P
	2	Determines if more than 1 money instance can be created/added	creates 3 new instances of PHP 5 (PHP 5 count = 6)	PHP 5 count = 9	PHP 5 count = 9	P
	3	Determines if a money instance is added when the denomination is not valid	add an instance of 220 denomination	throw illegalStateException	throw illegalStateException	p
getTotalMoney()	1	Determines if the total money computation is correct	all the denominations (1, 5, 10, 20, 50, 100, 200, 500, 1000) has a count of 10	total money = 18860	total money = 18860	P
setChange()	1	Determines if the change money are properly cleared (all instances are removed) and sets it to 0	-	-	-	P
RegularVM Class						

dispenseItem()	1	Determines if an item can be dispensed when the payment is successful and there is enough stocks	-	dispensing of item and stocks (item instance) will be decreased	dispensing of item and stocks (item instance) will be decreased	P
	2	Determines if an item can be dispensed if the payment is not successful	-	no dispensing of item	no dispensing of item	P
	3	quantity to be bought is more than the stocks available	stocks left : 5; quantity to be bought : 10	display "NOT ENOUGH STOCKS! Please try again"	display "NOT ENOUGH STOCKS! Please try again"	P
setSlot()	1	Determines if an item can be added to the slots of the regular vending machine	add tamago twice (in separate slots)	0	0	P
	2	Determines whether an item can be added when the slots in the regular vending machine are already full.	-	0	0	p
	3	Determines if an item can be added if the number of stocks to be put is greated than the slot capacity of the regular vending machine	slot capacity = 10; item's stock = 20	0	0	p
	4	Determines if an item can be added if the number of stocks is less than 0	item's stock <= 0	0	0	p
	5	Determines if an item can be added if there's still a slot available and the input is less than the number of capacity	-	1	1	p
setCapacityPerSlot()	1	Determines if the slot capacity is set when the the input is <= 0	input = 0	0	0	p
	2	Determines if the slot capacity is set when the input is < 10	input = 5	0	0	p
	3	Determines if the slot capacity is set when the input is >= 10	input = 20	1	1	P
restockItem()	1	Determines if an item can be restocked properly if the it exceeds the slot capacity	slot capacity = 10; input > 10	restocking of item will not be allowed	restocking of item will not be allowed	P
	2	Determines if an item can be restocked properly if the input is less than or equal to the slot capacity	slot capacity = 10; input <= 10	restocked properly	restocked properly	P
isEmpty()	1	Determines if the vending machine is empty	-	TRUE	TRUE	P
	2	Determines if the vending machine is not empty	-	FALSE	FALSE	P
emptySlots()	1	Clears/Removes all the items in the slots	-	clears everthing and sets each index to null	clears everthing and sets each index to null	P
SpecialVM Class						
computeTotal()	1	Determines the total amount (from the selected items) to be paid by the user	ukokkei broth = 120; chashu pork = 150; noodles = 50	total amount : 320	total amount : 320	P

computeCal()	1	Determines the total calories (from the selected items) to inform the user	ukokkei broth Cal = 50; chashu pork Cal = 200; noodles Cal = 100	total Calories: 350	total Calories: 350	P
removeAllItem()	1	Removes all items in the cart	Cart contains : ukkokei broth, chashu pork, noodles	cart contains : --	cart contains : --	P
dispenseItem()	1	Determines if an item can be dispensed when the payment is successful and there is enough stocks	-	dispensing of item and stocks (item instance) will be decreased	dispensing of item and stocks (item instance) will be decreased	P
	2	Determines if an item can be dispensed if the payment is not successful	-	no dispensing of item	no dispensing of item	P
	3	quantity to be bought is more than the stocks available	stocks left : 5; quantity to be bought : 10	no dispensing of item	no dispensing of item	P
	4	Determines if an item can be dispensed when its not allowed to be sold separately	tonkatsu broth	no dispensing of item	no dispensing of item	P
	5	Determines if all the items in the cart can be dispensed properly when the payment is allowed and there's enough stocks	-	dispensing of item and stocks (item instance) will be decreased	dispensing of item and stocks (item instance) will be decreased	P
setSlot()	1	Determines if an item can be added to the slots of the regular vending machine	add tamago twice (in separate slots)	0	0	P
	2	Determines whether an item can be added when the slots in the regular vending machine are already full.	-	0	0	p
	3	Determines if an item can be added if the number of stocks to be put is greated than the slot capacity of the regular vending machine	slot capacity = 10; item's stock = 20	0	0	p
	4	Determines if an item can be added if the number of stocks is less than 0	item's stock <= 0	0	0	p
	5	Determines if an item can be added if there's still a slot available and the input is less than the number of capacity	-	1	1	p
setCapacityPerSlot()	1	Determines if the slot capacity is set when the the input is <= 0	input = 0	0	0	p
	2	Determines if the slot capacity is set when the input is < 10	input = 5	0	0	p
	3	Determines if the slot capacity is set when the input is >= 10	input = 20	1	1	P
restockItem()	1	Determines if an item can be restocked properly if the it exceeds the slot capacity	slot capacity = 10; input > 10	restocking of item will not be allowed	restocking of item will not be allowed	P
	2	Determines if an item can be restocked properly if the input is less than or equal to the slot capacity	slot capacity = 10; input <= 10	restocked properly	restocked properly	P

	isEmpty()	1	Determines if the vending machine is empty	-	TRUE	TRUE	P
		2	Determines if the vending machine is not empty	-	FALSE	FALSE	P
	emptySlots()	1	Clears/Removes all the items in the slots	-	clears everthing and sets each index to null	clears everthing and sets each index to null	P
	Slot Class						
	decreaseItems()	1	amount bought is less than the stocks available	item has 10 stocks; amount bought is 1	item instance is decreased/removed. The count now will be 9	item instance is decreased/removed. The count now will be 9	P
		2	amount bought is negative	---	the program will not reach this input because it was already prevented earlier	the program will not reach this input because it was already prevented earlier	P
		3	amount bought is 0	---	the program will not reach this input because it was already prevented earlier	the program will not reach this input because it was already prevented earlier	P
	restock()	1	amount to be added is a positive number	item has 0 stocks; amount added is 15	numOfItems will be 15, and startInventory will also be 15	numOfItems will be 15, and startInventory will also be 16	P
		2	amount to be added is a negative number	---	the program will not reach this input because it was already prevented earlier	the program will not reach this input because it was already prevented earlier	P