Geslani & Marcelino

Class: `ProductModel Class`

Method	#	Test Description	Sample Input Data	Expected Output	Actual Output	P/F
`isExpired()`	1	Determines that a product with future expiration date is not expired		expirationDate = LocalDate.of(2026, 12, 31)	current date = 2025-11-26	P
	2	Determines that a product with past expiration date is expired		expirationDate = LocalDate.of(2024, 1, 1)	current date = 2025-11-26	P
	3	Determines that a product expiring today is not expired		expirationDate = LocalDate.of(2025, 11, 26)	current date = 2025-11-26	P
`reduceStock(int quantity)`	4	Reduces stock when quantity is less than current stock		quantityInStock = 50, quantity = 10	quantityInStock = 40	P
	5	Reduces stock to exactly zero when quantity equals stock		quantityInStock = 25, quantity = 25	quantityInStock = 0	P
`restock(int quantity)`	6	Increases stock when restocking with positive quantity		quantityInStock = 20, quantity = 30	quantityInStock = 50	P
	7	Stock remains unchanged when restocking with zero quantity		quantityInStock = 15, quantity = 0	quantityInStock = 15	P
	8	Increases stock significantly when restocking with large quantity		quantityInStock = 5, quantity = 1000	quantityInStock = 1005	P
Class: ShoppingCartModel Class						
`addItem	9	Adds new item to cart when product		prod = Coke (quantityInSto	prod = Coke (quantityInStoc	P

`(ProductModel prod, int quantity)``		has sufficient stock		ck=50), quantity = 5	k=50), quantity = 5	
	10	Rejects adding item when quantity exceeds available stock		prod = Skyflakes (quantityInStock=5), quantity = 10	Error message: "Insufficient stock!", items unchanged	P
	11	Rejects adding expired product to cart		prod with expirationDate = 2024-01-01, quantity = 3	Error message: "Cannot add expired product!", items unchanged	P
	12	Updates quantity when adding more of existing item within stock limit		prod = Coke (quantityInStock=50, currently in cart=10), quantity = 5	items contains Coke with quantity 15, success message printed	P
`` `calculateSubtotal()` ``	13	Calculates subtotal correctly with single item in cart		items contains: Coke (price=₱65.0) quantity=2	₱130.00	P
	14	Calculates subtotal correctly with multiple different items		items contains: Coke quantity=2 (₱130), Spam (price=₱185.0) quantity=1	₱315.00	P
	15	Returns zero when cart is empty		items is empty	₱0.00	P
`` `removeItem(ProductModel prod)` ``	16	Removes item successfully when product exists in cart		items contains Coke quantity=5, prod = Coke	items no longer contains Coke, message: "Removed from cart: Coke 1.5L"	P
	17	Shows error when trying to remove non-existent product		items contains Spam, prod = Coke	Error: "Product not found in cart!", items unchanged	P

	18	Shows error when trying to remove from empty cart		items is empty, prod = any product	Error: "Product not found in cart!", items remains empty	P
`getQuantity(ProductModel prod)`	19	Returns correct quantity for existing item in cart		items contains Coke quantity=7, prod = Coke	7	P
	20	Returns zero for product not in cart		items contains Spam quantity=2, prod = Coke	0	P
	21	Returns zero when cart is empty		items is empty, prod = any product	0	P
Class: MembershipCardModel Class						
`addPoints(double totalAmount)`	22	Adds points correctly when purchase amount equals exact multiple of 50		points = 0, totalAmount = 500.0	points = 10	P
	23	Does not add points when purchase amount is below threshold		points = 5, totalAmount = 25.0	points = 5	P
	24	Adds points correctly and accumulates with existing points		points = 20, totalAmount = 150.0	points = 23	P
`redeemPoints(double totalAmount)`	25	Redeems points and reduces total amount correctly		points = 100, totalAmount = 500.0	totalAmount = 490.0, points = 0	P
	26	Returns unchanged amount when no points available to redeem		points = 0, totalAmount = 200.0	totalAmount = 200.0, points = 0	P
Class TransactionModel Class						
`calculateTotal()`	27	Calculates total correctly with VAT and no discount		cart subtotal = 100.0, discount = 0	vat = 12.0, totalAmount = 112.0	P

	28	Calculates total correctly with both VAT and discount applied		cart subtotal = 100.0, discount = 20.0	vat = 12.0, totalAmount = 92.0	P
	29	Sets total to zero when discount exceeds subtotal plus VAT		cart subtotal = 50.0, discount = 100.0	totalAmount = 0.0	P
`applySeniorDiscount()`	30	Applies 20% senior discount and removes VAT for senior customer		customer.isSenior = true, cart subtotal = 1000.0	discount = 200.0, vat = 0.0, totalAmount = 800.0	P
	31	Does not apply discount for non-senior customer		customer.isSenior = false, cart subtotal = 1000.0	Error message printed, discount unchanged, vat = 120.0	P
	32	Applies senior discount correctly on small purchase amount		customer.isSenior = true, cart subtotal = 50.0	discount = 10.0, vat = 0.0, totalAmount = 40.0	P
`applyMembershipDiscount()`	33	Applies 5% membership discount when customer has points		customer.membership.points = 50, cart subtotal = 1000.0	discount increases by 50.0	P
	34	Does not apply discount when customer has no points		customer.membership.points = 0, cart subtotal = 500.0	Error message: "No membership points available", discount unchanged	P
	35	Applies membership discount correctly on small purchase		customer.membership.points = 10, cart subtotal = 100.0	discount increases by 5.0	P
`processPayment(double	36	Processes payment successfully when amount paid equals		totalAmount = 250.0, amountPaid = 250.0	change = 0.0, return true	P

amountPaid)`		total				
	37	Rejects payment when amount paid is less than total		totalAmount = 500.0, amountPaid = 300.0	Error message printed, return false	P
	38	Processes payment and calculates correct change when overpaid		totalAmount = 150.0, amountPaid = 200.0	change = 50.0, return true	P
Class: InventoryModel Class						
`addProduct(ProductModel prod)`	39	Adds new unique product successfully to inventory		prod = new ProductModel("Pepsi", "Pepsi Co", "Beverages", 60.0, 40, LocalDate.of(2026,6,30))	Product added, products.size() increases by 1, success message printed	P
	40	Rejects adding duplicate product with same name		prod = ProductModel with name "Coke 1.5L" (already exists)	Error: "Product already exists in inventory!", products.size() unchanged	P
	41	Adds product even when name is empty string		prod = new ProductModel("", "Brand", "Food", 10.0, 5, date)	Product added (edge case behavior)	P
`restockProduct(String name, int quantity)`	42	Increases stock correctly when restocking existing product		name = "Coke 1.5L" (current stock=50), quantity = 20	Product stock updated to 70, success message printed	P
	43	Shows error when trying to restock non-existent product		name = "NonExistentProduct", quantity = 10	Error message: "Product not found: NonExistentProduct"	P
	44	Keeps stock		name = "Spam	Product stock	P

		unchanged when restocking with zero quantity		Classic" (current stock=15), quantity = 0	remains 15	
`findProductByName(String name)`	45	Finds and returns existing product ignoring case		name = "coke 1.5l" (stored as "Coke 1.5L")	Returns ProductModel object for Coke 1.5L	P
	46	Returns null when product name does not exist		name = "Unknown Product"	Returns null	P
	47	Finds product correctly when name has extra whitespace		name = " Spam Classic "	Returns ProductModel object for Spam Classic	P
`getLowStockItems()`	48	Returns list of products with stock below threshold of 10		products: Eden Cheese (stock=8), Skyflakes (stock=5), Coke (stock=50)	List containing 2 products: Eden Cheese and Skyflakes	P
	49	Returns empty list when all products have sufficient stock		All products in inventory have quantityInStock >= 10	Empty ArrayList	P
	50	Returns empty list when inventory has no products		products list is empty	Empty ArrayList	P
`deleteProduct(ProductModel product)`	51	Removes existing product from inventory successfully		product = existing Coke ProductModel object	Returns true, product removed from products list, success message printed	P
	52	Returns false when trying to delete non-existent product		product = new ProductModel not in inventory	Returns false, products list unchanged, error message printed	P
	53	Returns false when		product list is	Returns false,	P

		trying to delete from empty inventory		empty, product = any ProductMode	error message printed	
`reduceStockAfterPurchase(ProductModel prod, int quantity)`	54	Reduces stock correctly when sufficient quantity available		prod = Coke (current stock=50), quantity = 10	Product stock reduced to 40, success message printed	P
	55	Shows error when trying to reduce more than available stock		prod = Eden Cheese (current stock=8), quantity = 15	Error: "Insufficient stock for: Eden Cheese", stock unchanged	P
	56	Shows error when trying to reduce stock of non-existent product		prod = ProductModel not in inventory, quantity = 5	Error: "Product not found in inventory", no changes made	P
Class: CategoryHelper Class						
`isValidCategory(String category)`	57	Returns true when category exists in predefined list		category = "Food"	true	P
	58	Returns false when category does not exist in list		category = "Electronics"	false	P
	59	Returns false when category has incorrect case		category = "food" (lowercase)	false	P
Class: CustomerModel Class						
`viewTotalCost()`	60	Returns correct total cost with multiple items in cart		shoppingCart contains: Coke x2 (₱130), Spam x1 (₱185)	₱315.00	P
	61	Returns zero when cart is empty		shoppingCart is empty	₱0.00	P
	62	Returns correct total cost with single item		shoppingCart contains: Nescafe x10 (₱8 each)	₱80.00	P

`addToCart(ProductModel prod, int quantity)`	63	Adds valid product to shopping cart successfully		prod = Coke (quantityInStock=50), quantity = 3	shoppingCart.items contains Coke with quantity 3	P
	64	Delegates validation to ShoppingCartModel for zero quantity		prod = Spam, quantity = 0	Behavior depends on ShoppingCartModel.addItem validation	P
	65	Delegates validation to ShoppingCartModel for exceeding stock		prod = Eden Cheese (quantityInStock=8), quantity = 20	ShoppingCartModel.addItem handles validation and rejects	P