Unit Test (Set One) - Parleen

## Target component

ItemBox. We verify: addItem(id), removeItem(id), getItem(), getItem(id), and count().

## Test fixtures

- ItemBox(name="T-Shirt", color="Red", size="M")
- Start with quantity = 2 and items = {101: Item(...id=101), 102: Item(...id=102)}.
- Fresh Item objects for IDs 103, 104 as needed. (No Inventory/Register involved.)

## Tests

1. **Add item increases quantity and map**
   Pre: quantity=2, ids {101,102}.
   Action: addItem(103)
   Expect: count()==3; items contains 103; other entries unchanged.
2. **Remove existing item decreases quantity and map**
   Pre: quantity=3 with ids {101,102,103}.
   Action: removeItem(102)
   Expect: count()==2; id 102 removed; 101 and 103 remain.
3. **Remove non-existent id is rejected and side-effect free**
   Pre: quantity=2 with ids {101,103}.
   Action: removeItem(999)
   Expect: throws NoSuchElementException (or your designated error type); count() and items unchanged.
4. **getItem(id) returns exact object; no mutation**
   Pre: quantity=2 with ids {101,103}.
   Action: getItem(101)
   Expect: returned object has id==101; count() unchanged; map unchanged.
5. **getItem() returns some available item when non-empty**
   Pre: quantity=2.
   Action: x = getItem()
   Expect: x is one of {101,103}; count() unchanged.
6. **getItem() on empty box is safe**
   Pre: quantity=0; items={}.
   Action: getItem()
   Expect: returns null (or throws a well-documented error). Document whichever contract you implement, but the test asserts **no mutation** of state.
7. **Duplicate add is rejected**
   Pre: quantity=2 with ids {101,103}.
   Action: addItem(101)
   Expect: throws IllegalArgumentException (or your chosen error); state unchanged.
8. **Quantity consistency after mixed operations**
   Pre: empty box.
   Action: addItem(201), addItem(202), removeItem(201)
   Expect: count()==1 and only id 202 present.

Unit Test (Set Two) - Adarsh

## Target component

Register **cart management only**: addToCart(item), removeFromCart(id), removeFromCart(name), viewCart(), clearCart() .

## Test doubles

- Use a stub Inventory (or null) because these methods operate on the internal cart list per your design.
- Important contract from your SDS/System tests: adding to cart must **not** modify inventory until a purchase is completed. We assert no inventory interaction here. Software Design Specifications

## Test fixtures

- Fresh Register with cart=[].
- Items:
    - shirtA = Item(name="T-Shirt", id=501, price=20.00, tax=1.60, color="Red", size="M")
    - shirtB = Item(name="T-Shirt", id=502, price=22.00, tax=1.76, color="Red", size="L")
    - jeans = Item(name="Jeans", id=601, price=40.00, tax=3.20, color="Blue", size="32")

## Tests

1. **addToCart appends one item; viewCart reflects it**
   Pre: empty cart.
   Action: addToCart(shirtA)
   Expect: viewCart() length 1; element has id=501; no inventory calls.
2. **Adding multiple items preserves all entries (order acceptable)**
   Pre: cart has shirtA.
   Action: addToCart(jeans)
   Expect: viewCart() contains ids {501,601}; sizes and names intact.
3. **removeFromCart(id) removes only the matching item**
   Pre: cart has shirtA, jeans.
   Action: removeFromCart(501)
   Expect: cart now only has jeans; no effect on totals/cash/receipts (those belong to makeTransaction).
4. **removeFromCart(id) with unknown id is safe**
   Pre: cart has jeans only.
   Action: removeFromCart(999)
   Expect: cart unchanged; no exception (or throws documented NoSuchElementException—pick one contract and keep it consistent).
5. **removeFromCart(name) removes exactly one matching item when duplicates exist**
   Pre: cart has shirtA and shirtB (same name, different IDs/sizes).
   Action: removeFromCart("T-Shirt")
   Expect: cart size decreases by 1; exactly one of {501,502} removed; the other remains.
6. **clearCart empties the cart**
   Pre: cart has 1+ items.

Action: clearCart()
Expect: viewCart() is empty list.

7. **Cart operations do not touch inventory**
   Pre: substitute a spy/stub inventory on Register.
   Action: run tests 1–6.
   Expect: zero calls to inventory-mutating methods (removeItem, addItem, updateInventory). This enforces the SDS rule that inventory updates occur at transaction time, not during cart ops. Software Design Specifications
8. **Idempotence of clearCart**
   Pre: cart already empty.
   Action: clearCart() again.
   Expect: still empty; no errors.

Integration Test (Set One) - Jason

Feature: `makeTransaction(cash: double, card_digits: integer): void`

**1. One item transaction**

Precondition: cash is currently zero and with only one item in cart
Action: call makeTransaction(cash, card_digits) with one item in cart
Expected: cash should equal to the one item's price, generate a new receipt, inventory cleared.

**2. Multiple items transaction**

Precondition: cash is currently zero with multiple item in cart
Action: call makeTransaction(cash, card_digits) with multiple items in cart
Expected: cash should equal to all items' price in the cart combined, generate a new receipt

**3. Multiple transactions**

Precondition: cash is currently zero
Action: call makeTransaction(cash, card_digits) one time with one item in cart and another time, with another different item in cart
Expected: cash should equal the two items' price combined, generate two new receipts, inventory cleared.

**4. Invalid card digits**

Precondition: cash is currently zero
Action: call makeTransaction(cash, card_digits) with a invalid card digits
Expected: operation fails with error message of invalid card digits, cash remains zero, inventory remains.

**5. Tax inclusion**

Precondition: cash is currently zero and with only one item in cart with tax
Action: call makeTransaction(cash, card_digits) with only one item in cart with tax
Expected: cash should equal to the one item's price with tax added, generate a new receipt, inventory cleared

**6. Add verify**

Precondition: cash is currently zero and with no item in cart
Action: call addToCart(item) to add an item and call makeTransaction(cash, card_digits)
Expected: cash should equal to the item's price, generate a new receipt, inventory cleared

**7. Remove verify**
Precondition: cash is currently zero and with only two different items in cart
Action: call removeFromCart(name) to remove the second item and call makeTransaction(cash, card_digits)
Expected: cash should equal to the first item's price, generate a new receipt, inventory cleared
**8. Clear Cart verify**
Precondition: cash is currently zero and with three different items in cart
Action: call clearCart() to remove the items and call makeTransaction(cash, card_digits)
Expected: operation fails with error message of inventory empty, cash remains zero, inventory empty.

Integration Test (Set Two) - Caleb
Feature: refundTransaction(id: integer): void
**1. Full refund - Cash Sale**
Precondition: A cash sale of 2 shirts is completed.
Action: Call refundTransaction(id) with the receipt ID.
Expected: Inventory quantity of Item A increases by 2. Register cash decreases by the refunded total. totalSales decreases by the same amount. Receipt remains accessible in the register's transaction list.
**2. Multiple items refund**
Precondition: A sale includes 3+ different items.
Action: Refund the entire receipt.
Expected: Each item's quantity is restored. Cash and totalSales adjusted by full combined total.
**3. Invalid Transaction ID**
Action: Attempt to refund a non-existent ID.
Expected: Operation fails with clear error message and no changes to inventory, cash, or totalSales.
**4. Duplicate Refund Attempt**
Precondition: Transaction X already refunded.
Action: Call refundTransaction(X) again.
Expected: Second call rejected and there are no changes to inventory or cash.
**5. Partial Failure Handling**
Simulation: Force inventory update failure mid-refund
Expected: Transaction not marked as refunded and cash and inventory remains unchanged.
**6. Shopping Car Isolation**
Precondition: Active cart contains items for a new sale.
Action: Refund a previous transaction.
Expected: Cart remains unchanged and only past inventory and cash affected.
**7. Tax Verification**
Precondition: Sale includes tax.
Action: Refund the sale.
Expected: Refunded amount equals receipt total (subtotal + tax) and tax portion correctly reversed in totalSales.
**8. Insufficient Cash in Drawer**

Precondition: Cash on hand is less than the refund amount.
Expected: Refund rejected.

**9. Audit and Record Integrity**

Action: Verify logs after refund.
Expected: Refund entry recorded with same ID and timestamp and no accidental deletion of original receipt data.

System Test (Set One) - Drew
Feature: Inventory

**1. Add a new item to inventory**

The worker adds a new item with a unique ID and complete details to the inventory. The system creates a new entry and displays the item in the inventory list with the correct attributes and quantity.

**2. Add an item with the same ID (should not work)**

The worker tries to add a new item using an ID that already exists in the system. The system rejects the addition and displays an error message indicating that the item ID is already in use.

**3. Add an item with the same attributes as another item in the inventory**

The worker adds an item that has the same name, color, and size as an existing item but a different ID. The system updates the existing ItemBox and increases the total quantity for that item type.

**4. Remove an existing item**

The worker selects an item by its valid ID and removes it from the inventory. The system decreases the quantity of that item by one, or removes it entirely if the quantity reaches zero.

**5. Remove an item that does not exist**

The worker attempts to remove an item using an ID that is not in the inventory. The system does not change any inventory data and displays an error message indicating the item ID was not found.

**6. Search for an item by ID**

The worker searches the inventory using a specific item ID. The system displays the exact item that matches the ID along with its full details.

**7. Search for an item by attributes**

The worker searches the inventory using one or more attributes such as name, color, or size. The system lists all items that match those attributes and shows their IDs, quantities, and details.

System Test (Set Two) - Nathan
Feature: Register

**1) Retrieving a receipt**

- An employee searches for a specific transaction via an ID
- The system locates and returns the receipt with its items, total cost, tax, payment method, and date of the transaction

**2) Refunding a whole transaction**

- A customer wants to return all items via a previous buy
- The system restores all items to inventory, and in addition issues a full refund, which will update the total sales and cash amounts

**3) Removing a transaction from the system**

- An employee deletes a transaction record from the system using an ID
- The record is removed and does not change any information in regards to the financial or inventory data

**4) Refunding a singular item from a transaction**

- A customer wants to return a single item from a purchase that included multiple items
- The system restores the returned item into inventory, using its given ID. Alongside this, a cash refund is done for the customer.

**5) Cancelling of a purchase before payment**

- The customer decides to stop the purchase
- The system clears the potential buy, leaving the inventory and financial data in place

**6) Adding an item to the cart**

- An employee scans a given item and adds it to the current shopping cart
- The item appears in the in the cart list, ready to be bought, while the inventory does not adjust until the purchase is complete

**7) Removing an item from the cart**

- A customer wants to remove one specific item from the cart that has already been scanned
- The item is removed from the cart list using its ID, with the inventory and total sales amount to change soon afterwards.

**8) Removing an item from the cart using its name**

- The employee removes one out of two products that share the same name
- The system is given the name and selects the specific variation of the item to be removed from the cart list

**9) Viewing what items are in the cart**

- Before checking out, the employee reviews the shopping cart list and its entities
- The system displays all selected items with their given attributes some of which include its name, ID, price, and tax

**10) Completing the purchase using either cash or credit card (digits)**

- The customer plans to complete the payment process using either cash or card

- The system finalizes the selling, creates a receipt, adjusts the cash amount (if applicable), updates the total sales amount, modifies inventory, and the cart before eventually recording the transaction.