## Anvils 1.1

Version 1.1

Written by Oscar Moreno

Edited by Sarah Murphree

Date: 03-19-2025

## Table of Contents

Version 1.1

Business Statement/Functional Narrative/Scope .......................................................3

Scope.......................................................................................................................3

Assumptions............................................................................................................3

List of Steps .............................................................................................................4

User Stories..............................................................................................................6

Use Case Scenarios..................................................................................................10

Test Plan ...............................................................................................................17

Technical Requirements .........................................................................................22

Entity Relationship Diagram ....................................................................................23

Class Diagram ........................................................................................................24

Sequence Diagram .................................................................................................25

Class Responsibility Diagram ..................................................................................26

Task Object Responsibility ......................................................................................27

Activity Flow ...........................................................................................................28

State Diagram ........................................................................................................29

Test Cases .............................................................................................................30

Executable Tests ....................................................................................................36

Revision History .....................................................................................................54

Review ...................................................................................................................55

Version 1.0 .............................................................................................................57

## Business Statement / Functional Narrative / Scope

**What does the system do:**  
The ACME Anvils Order Taking app lets a clerk take a single order for anvils from a customer. It starts by asking the clerk’s name, shows a welcome message, collects customer shipping info, loyalty club status, and anvil quantity, calculates the cost (with tiered pricing, tax, applicable discounts, and shipping), and prints a neatly formatted invoice. It’s built for one order per run—clerks restart it for each new order. This tool keeps ACME’s cartoon villain customers equipped with anvils, maintaining their 50+ year legacy.

Scope

* Collect the clerk’s name and customer order details.
* Calculate costs (subtotal, applicable discounts for loyalty club, quantity ordered, & state shipped to, taxable amount, tax, shipping) and display an invoice matching the sample output.
* Handle one order per execution with pauses before invoice and exit.

## Assumptions

* The clerk enters valid data (no validation required for Version 1 & 1.1).
* Console environment supports basic formatting (spaces, newlines).
* Pricing: $88.50 (1–9), $70.00 (10–19), $68.25 (20+); Tax: 9.5%; Shipping: $5/lbs. (anvils are 50lbs)
* Discounts: Shipping is discounted 10% if 6+ anvils are ordered from California or Oregon; Orders with 10+ anvils ordered are discounted 10%; If a customer is a loyalty club member they are given a 15% discount.
* Sample output formatting supersedes minor spec inconsistencies (e.g., prompt wording).

List of Steps:

1. **System Launch**
   1. The system is started, and the order-taking session is initialized.
2. **Display Welcome Message**
   1. The system displays a welcome message along with the ACME slogan and branding (including stars).
3. **Prompt for Clerk's Name**
   1. The system asks the clerk to enter their name.
   2. The clerk enters their name, which will later appear on the invoice.
4. **Prompt for Anvil Order Quantity**
   1. The system asks for the number of anvils the customer wants to order.
   2. The clerk enters a valid numeric quantity.
5. **Prompt for Loyalty Club Membership**
   1. The system asks whether the customer is a loyalty club member with a prompt like “Are you a member of our Futility Club Loyalty program? (Y)”.
   2. The clerk enters “Y” or “y” if the customer is a member.
6. **Prompt for Customer Shipping Details**
   1. The system requests the customer's shipping information: name, address, city, state, and zip code.
   2. The clerk enters the shipping details.
7. **Determine Pricing**
   1. The system uses tiered pricing based on the number of anvils ordered:
      1. 1–9 anvils: $88.50 each
      2. 10–19 anvils: $70.00 each (apply 10% discount on orders of 10+)
      3. 20+ anvils: $68.25 each (apply 10% discount on orders of 10+)
   2. The appropriate per-anvil price is determined based on the entered quantity.
8. **Calculate Subtotal**
   1. Multiply the per-anvil price by the quantity ordered to compute the subtotal.
9. **Apply Loyalty Discount (if applicable)**
   1. If the customer is a loyalty club member, a 15% discount is applied to the subtotal.
   2. The system calculates the discount amount and adjusts the taxable amount accordingly.
10. **Check for Shipping Discount**
    1. If more than 5 anvils are ordered and the customer’s state is either California (CA) or Oregon (OR), a 10% discount is applied to the shipping cost.
    2. Otherwise, standard shipping rules apply.
11. **Calculate Shipping Cost**
    1. Shipping cost is calculated based on weight ($5 per lbs) with each anvil weighing 50 lbs.
    2. The total shipping cost is determined by multiplying the shipping cost per anvil by the number of anvils and then applying any eligible discount.
12. **Calculate Taxable Amount and Sales Tax**
    1. The taxable amount is calculated (typically the subtotal minus any loyalty discount, if applicable).
    2. Sales tax is applied at a rate of 9.5% on the taxable amount.
13. **Calculate Total Cost**
    1. The final total is determined by adding the subtotal, sales tax, and shipping cost together.
14. **Invoice Display Prompt**
    1. The system prompts the clerk to press a key to display the final invoice.
15. **Generate and Display Invoice**
    1. The system generates a neatly formatted invoice that includes:
       1. Cost per anvil
       2. Subtotal
       3. Any discounts (loyalty and/or quantity-based)
       4. Taxable amount
       5. Sales tax
       6. Shipping cost
       7. Total order cost
    2. If the customer is a loyalty club member, the invoice shows additional lines such as “Less 15% Loyalty Club” and recalculates the taxable amount.
16. **Display Promotional Message**
    1. After the invoice, the system displays a promotional message thanking the customer (with the ACME slogan).
17. **End Session**
    1. The system prompts the clerk to press a key to end the program.
    2. The order-taking session is terminated, allowing for a new order to be started if needed.

## User Stories

**US1 - As a clerk**, I want to enter my name, so it appears on the invoice.

**US2 - As a clerk**, I want a welcome message with stars and the motto for branding.

**US3 - As a clerk**, I want to input order details with specific prompts on the same line as responses.

**US4 - As a clerk,** I want to ask the customer if they are a member of the loyalty club.

**US5** –**As a clerk,** I want to the invoice to calculate the invoice to include applicable discounts.

**US6 - As a clerk**, I want an invoice showing all costs, loyalty member discount (if applicable), and shipping details in a boxed format.

**US7 - As a clerk**, I want pauses to control when the invoice appears, and the program ends.

|  |  |  |
| --- | --- | --- |
| **Step** | **Description** | **User Story(ies) Addressed** |
| **1. System Launch** | The system starts and the order-taking session is initialized. | (System readiness, implicit to all use cases) |
| **2. Display Welcome Message** | A welcome message with stars and the ACME slogan is displayed. | US2 – Welcome message and branding |
| **3. Prompt for Clerk's Name** | The system asks for the clerk's name, which will later appear on the invoice. | US1 – Enter clerk’s name |
| **4. Prompt for Anvil Order Quantity** | The system asks for the number of anvils the customer wants to order, and the clerk enters a valid number. | US3 – Input order details |
| **5. Prompt for Loyalty Club Membership** | The system asks if the customer is a loyalty club member (Y/N). | US4 – Ask about loyalty membership |
| **6. Prompt for Customer Shipping Details** | The system collects the customer’s shipping information (name, address, city, state, zip). | US3 – Input order details |
| **7. Determine Pricing** | Based on the number of anvils ordered, tiered pricing is applied (including any quantity-based discount). | US5 – Calculate discounts (quantity discount) |
| **8. Calculate Subtotal** | Multiply the per-anvil price by the number of anvils to compute the subtotal. | US5 – Invoice calculation |
| **9. Apply Loyalty Discount (if applicable)** | If the customer is a loyalty club member, apply a 15% discount and adjust the taxable amount. | US5 – Include loyalty discount in calculations |
| **10. Check for Shipping Discount** | Determine if the shipping cost qualifies for a 10% discount (if >5 anvils are ordered from CA/OR). | US5 – Apply shipping discount |
| **11. Calculate Shipping Cost** | Calculate shipping cost based on weight (50 lbs per anvil at $5/lb), then apply any discount determined. | US5 – Invoice calculation |
| **12. Calculate Taxable Amount and Sales Tax** | Calculate the taxable amount (adjusting for any discounts) and then apply a 9.5% sales tax. | US5 – Invoice calculation |
| **13. Calculate Total Cost** | Sum the subtotal, sales tax, and shipping to compute the final total cost. | US5 – Invoice calculation |
| **14. Invoice Display Prompt** | Prompt the clerk to press a key when ready to display the invoice. | US7 – Control when the invoice appears |
| **15. Generate and Display Invoice** | Generate a neatly formatted invoice showing the cost per anvil, subtotal, discounts (loyalty, quantity), tax, shipping, and total in a boxed format. | US6 – Invoice display; US7 – Invoice appearance and control |
| **16. Display Promotional Message** | Display a thank-you or promotional message with the ACME slogan after the invoice is shown. | US2 – Branding and promotional messaging |
| **17. End Session** | The system prompts the clerk to press a key to end the session, after which the process terminates (allowing restart for a new order). | US7 – End the program |

Acceptance Criteria:

**ACUS1**

1.1 - The system displays a prompt asking for the clerk’s name as the first step in the order process.

1.2 - The clerk can enter any valid string as their name.

1.3 - The invoice output clearly shows the clerk’s name in the correct location.

**ACUS2**

2.1 - Immediately upon launch, the system displays a welcome message before any further input is requested.

2.2 - The welcome message includes decorative elements (such as stars) and the ACME motto/slogan.

2.3 - The branding elements (stars and motto) are clearly visible and formatted as specified by the sample output.

2.4 - The welcome message appears consistently for each order session.

**ACUS3**

3.1 - The system provides a prompt that appears on the same line where the clerk types the response (e.g., “Enter number of anvils:” with the input cursor immediately following the colon).

3.2 - All order-related inputs (such as anvil quantity and customer shipping details) use this single-line prompt format.

3.3 - The prompts match the wording and format described in the requirements, ensuring consistency with sample output.

3.4 - The system accepts valid responses following the prompt without requiring additional formatting adjustments.

**ACUS4**

4.1 - The system displays a clear prompt such as “Are you a member of our Futility Club Loyalty program? (Y/N)” after entering order details.

4.2 - The prompt accepts “Y” or “y” for yes.

4.3 - Based on the input, the system correctly marks the customer’s loyalty club status.

4.4 - If the customer is a loyalty club member, an appropriate confirmation (e.g., a message like “You'll get that AMAZING 15% loyalty bonus today!”) is displayed.

**ACUS5**

5.1 - The system determines the per-anvil price based on quantity tiers:

5.1.1 - 1–9 anvils: $88.50 each

5.1.2 - 10–19 anvils: $70.00 each (with a 10% discount applied on the base tier, if specified)

5.1.3 - 20+ anvils: $68.25 each (with a 10% discount applied on the base tier)

5.2 - If the customer is a loyalty club member, an additional 15% discount is applied to the subtotal.

5.3 - For orders where more than 5 anvils are ordered and the customer is located in CA or OR, a 10% discount on shipping is applied.

5.4 - Calculations include:

5.4.1 - Subtotal (price per anvil × quantity)

5.4.2 - Loyalty discount (15% off subtotal, if applicable)

5.4.3 - Taxable amount (subtotal minus loyalty discount, if applicable)

5.4.4 - Sales tax (9.5% applied to the taxable amount)

5.4.5 - Shipping cost (based on weight and any shipping discounts)

5.4.6 - Final total (sum of subtotal, tax, and shipping)

5.5 All computed amounts must match the formulas.

**ACUS6**

6.1 - The final invoice displays:

6.1.1 - Cost per anvil

6.1.2 - Subtotal

6.1.3 - Any applicable discounts (e.g., “Less 15% Loyalty Club:” if the customer is a member)

6.1.4 - Taxable amount

6.1.5 - Sales tax

6.1.6 - Shipping cost

6.1.7 - Final total

6.2 - The invoice includes the customer’s shipping information (name, address, city, state, zip).

6.3 - The invoice is formatted in a clear, boxed layout that matches the sample output.

6.4 - All numerical values are formatted correctly (e.g., two decimal places) and reflect the accurate calculations.

**ACUS7 –**

7.1 - After completing all data entry and calculations, the system prompts the clerk to “Press any key to display the invoice.”

7.2 - The system pauses and waits for user input before displaying the invoice.

7.3 - Once the invoice is shown, the system prompts the clerk to “Press any key to end the program” (or similar), pausing again before termination.

7.4 - The pause functionality is reliably triggered at both points so the clerk can control the flow of the session.

## Use Case Scenarios

**Use Case 1: Placing a Standard Order**

**User Story:** US6

**Acceptance Criteria:** ACUS6

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched, and the clerk is ready to take an order.

**Main Flow:**

1. The system prompts the clerk for their name.
2. The system displays a welcome message with the ACME slogan.
3. The system asks for the number of anvils to order.
4. The clerk enters a valid number of anvils.
5. The clerk asks if the customer is a loyalty club member.
6. The clerk enters “Y” or “y” if the customer is a loyalty club member.
7. The system requests the customer's shipping details (name, address, city, state, zip).
8. The clerk enters the customer’s information.
9. The system calculates the total cost, including tax, loyalty club discount, and shipping.
10. The system prompts the clerk to press a key to display the invoice.
11. The system displays the invoice with all order details.
    1. If the customer is a loyalty member the additional lines are included in the invoice amount section:
       1. “Less 15% Loyalty Club: ($##.##)
       2. Taxable Amount: $##.##
12. The system thanks the user with a promotional slogan.
13. The system prompts the clerk to press a key to end the program.

**Postconditions:**

* The order details are displayed successfully.
* The clerk is prompted to restart a new order.

**Use Case 2: Ordering Greater Than 5 Anvils in CA or OR (Discounted Shipping)**

**User Story:** US5

**Acceptance Criteria:** ACUS5

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched, and the clerk is ready to take an order.
* The customer is located in **California (CA) or Oregon (OR)**.
* The customer orders greater than 5 anvils.

**Main Flow:**

1. The clerk enters a valid number of anvils (>5).
2. The clerk provides the customer’s shipping address, including **CA or OR** as the state.
3. The system calculates the total cost of shipping with a 10% discount.
4. The system calculates the total cost, including tax and shipping.
5. The invoice is displayed with cost per anvil, subtotal, sales tax, discounted shipping, and total.
6. The system thanks the user and ends the session.

**Postconditions:**

* The invoice reflects the discounted shipping.
* The order total is correctly calculated.

**Use Case 3: Ordering 10+ Anvils for Discounted Pricing**

**User Story:** US3

**Acceptance Criteria:** ACUS3

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched, and the clerk is ready to take an order.
* The customer orders **10 or more anvils**.

**Main Flow:**

1. The clerk enters a number between 10 and 19.
2. The system applies to the price of $63.00 per anvil (the original price of $70.00 with a 10% discount for 10+ anvils ordered).
3. The system calculates subtotal, tax, and shipping fees.
4. The system generates the invoice with the discounted price per anvil.
5. The system displays the invoice and thanks the user.

**Postconditions:**

* The invoice reflects the correct discounted price.

**Use Case 4: Ordering 20+ Anvils for Bulk Discount**

**User Story:** US3

**Acceptance Criteria:** ACUS3

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched, and the clerk is ready to take an order.
* The customer orders **20 or more anvils**.

**Main Flow:**

1. The clerk enters 20 or more anvils.
2. The system applies to the price of$61.425 per anvil (the original price of $68.25 with a 10% discount for 10+ anvils ordered).
3. The system calculates subtotal, tax, and shipping fees.
4. The system generates the invoice with the bulk discount pricing.
5. The system displays the invoice and thanks the user.

**Postconditions:**

* The invoice reflects the correct bulk discount.

**Use Case 5: Invalid Data Entry (Optional - If Validation Is Implemented)**

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched.

**Main Flow:**

1. The clerk enters **an invalid number of anvils** (e.g., negative number, letters).
2. The system displays an **error message** and asks for a valid input.
3. The clerk enters **an invalid zip code** (e.g., non-numeric characters).
4. The system displays an **error message** and requests re-entry.
5. The system only proceeds once all inputs are valid.

**Postconditions:**

* The system handles errors properly and prevents invalid data submission.

**Use Case 6: Customer is a Futility Club Member (Apply Discount)**

**User Story:** US4

**Acceptance Criteria:** ACUS4

**Preconditions:**

* The system is launched, and the clerk is ready to take an order.

**Main Flow:**

* The system prompts the clerk for their name.
* The system displays a welcome message with the ACME slogan.
* The system asks for the number of anvils to order.
* The clerk enters a valid number of anvils.
* The clerk asks if the customer is a loyalty club member, “Are you a member of our Futility Club Loyalty program ("Y" if yes)?”
* The clerk enters “Y” or “y” if the customer is a loyalty club member.
* The system displays the message, “You'll get that AMAZING 15% loyalty bonus today!”
* The system requests the customer's shipping details (name, address, city, state, zip).
* The clerk enters the customer’s information.
* The system calculates the total cost, including tax, loyalty club discount, and shipping.
* The system prompts the clerk to press a key to display the invoice.
* The system displays the invoice with all order details.
  + the additional lines are included in the invoice amount section:
    - “Less 15% Loyalty Club: ($##.##)
    - Taxable Amount: $##.##
* The system thanks the user with a promotional slogan.
* The system prompts the clerk to press a key to end the program.

**Postconditions:**

* The order details are displayed successfully.
* The clerk is prompted to restart a new order.

**Use Case 7: Clerk Enters Their Name**  
 **User Story:** US1

**Acceptance Criteria:** ACUS1*.*

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system has been launched and initialized for a new order.
* The system is ready to accept user input.

**Main Flow:**

1. **System Launch:** The order-taking session begins, and the system initializes the user interface.
2. **Prompt for Clerk’s Name:** The system displays a prompt (e.g., “Please enter your name:”) on the console.
3. **Input Clerk Name:** The clerk enters their name using the keyboard.
4. **Store and Display:** The system captures the entered name and stores it for later use. The clerk’s name is then made available to be included on the invoice header or designated area.
5. **Proceed to Next Step:** The system continues with subsequent order details after the name entry is successfully recorded.

**Alternative Flow (Optional):**

* If the clerk enters an invalid or empty string (when validation is implemented), the system displays an error message and re-prompts for a valid name.

**Postconditions:**

* The clerk’s name is successfully stored and will be displayed on the final invoice.
* The session advances to collecting further order details.

**Use Case 8: Display Welcome Message with Branding**  
**User Story:** US2

**Acceptance Criteria:** ACUS2

**Actors:**

* System
* Clerk (as the recipient of the message)

**Preconditions:**

* The system has launched and is in the initial state before any order information is entered.

**Main Flow:**

1. **System Launch:** Upon starting the order-taking application, the system immediately initiates the welcome sequence.
2. **Display Welcome Message:** The system outputs a welcome message that includes:
   1. Decorative elements (e.g., stars or similar motifs).
   2. The ACME motto or slogan for branding.
3. **Visibility & Readability:** The welcome message is formatted clearly and prominently, ensuring that the clerk sees the branding details before proceeding.
4. **Transition to Order Entry:** Once the welcome message is displayed, the system moves on to the next step (e.g., prompting for the clerk’s name).

**Alternative Flow:**

* If additional acknowledgments are required (e.g., the clerk must press a key to dismiss the message), the system pauses until the clerk confirms before proceeding.

**Postconditions:**

* The clerk has viewed the branded welcome message.
* The order-taking session continues with the clerk’s subsequent inputs.

## Test Plans & Supporting Bases

**1. Truth Tables (Logical Rules for Order Processing)**

A **Truth Table** represents different scenarios of discounted shipping eligibility.

|  |  |  |
| --- | --- | --- |
| **Anvils Ordered** | **State (CA/OR)** | **Free Shipping Applied?** |
| 1-4 | CA | No |
| 1-4 | OR | No |
| 1-4 | AZ | No |
| 6+ | CA | Yes |
| 6+ | OR | Yes |
| 10+ | Any State | No |

**2. Decision Table (Pricing and Tax Calculation Rules)**

**\****shipping cost per one anvil = $250 = $5\*50lbs*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Condition** | **1-9 Anvils** | **10-19 Anvils** | **20+ Anvils** | **10% discount to cost/anvil** | **Futility Club Member (15% discount)** | **CA/OR & >5 Anvils** | **Shipping Charge** | **Tax (9.5%) Applied?** |
| Scenario 1: Order 3 anvils in CA, isMember : True | $88.50 each | - | - | No | Yes | Yes | $225 per anvil | Yes |
| Scenario 2: Order 10 anvils in AZ, isMember : True | - | $70.00 each | - | Yes | Yes | No | $250 per anvil | Yes |
| Scenario 3: Order 25 anvils in TX, isMember : True | - | - | $68.25 each | Yes | Yes | No | $250 per anvil | Yes |
| Scenario 1: Order 3 anvils in CA, isMember : False | $88.50 each | - | - | No | No | Yes | $225 per anvil | Yes |
| Scenario 2: Order 10 anvils in AZ, isMember : False | - | $70.00 each | - | Yes | No | No | $250 per anvil | Yes |
| Scenario 3: Order 25 anvils in TX, isMember : False | - | - | $68.25 each | Yes | No | No | $250 per anvil | Yes |

**3. Boundary Value Analysis (BVA) Table**

**Testing boundary conditions for order quantity**:

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Input Value** | **Expected Outcome** |
| Min Order | 1 | Correct price tier applied |
| Just Below Discount | 9 | $88.50 per anvil |
| First Discount Threshold | 10 | $63.00 per anvil |
| Just Below Bulk Discount | 19 | $63.00 per anvil |
| Bulk Discount Threshold | 20 | $61.425 per anvil |
| Extreme Large Order | 1000 | System handles large numbers correctly |

**4. Equivalence Partitioning (EP) Table**

Dividing the order values into **equivalence classes** to test representative cases:

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition** | **Valid/Invalid** | **Test Input** | **Expected Result** |
| Small Order | Valid | 3 anvils | Price = $88.50 per anvil |
| Mid-Tier Order | Valid | 15 anvils | Price = $70.00 per anvil \* 10% discount |
| Bulk Order | Valid | 25 anvils | Price = $68.25 per anvil \* 10% discount |
| Invalid Negative Order | Invalid | -1 anvils | Error Message |
| Non-Numeric Input | Invalid | "ten" anvils | Error Message |

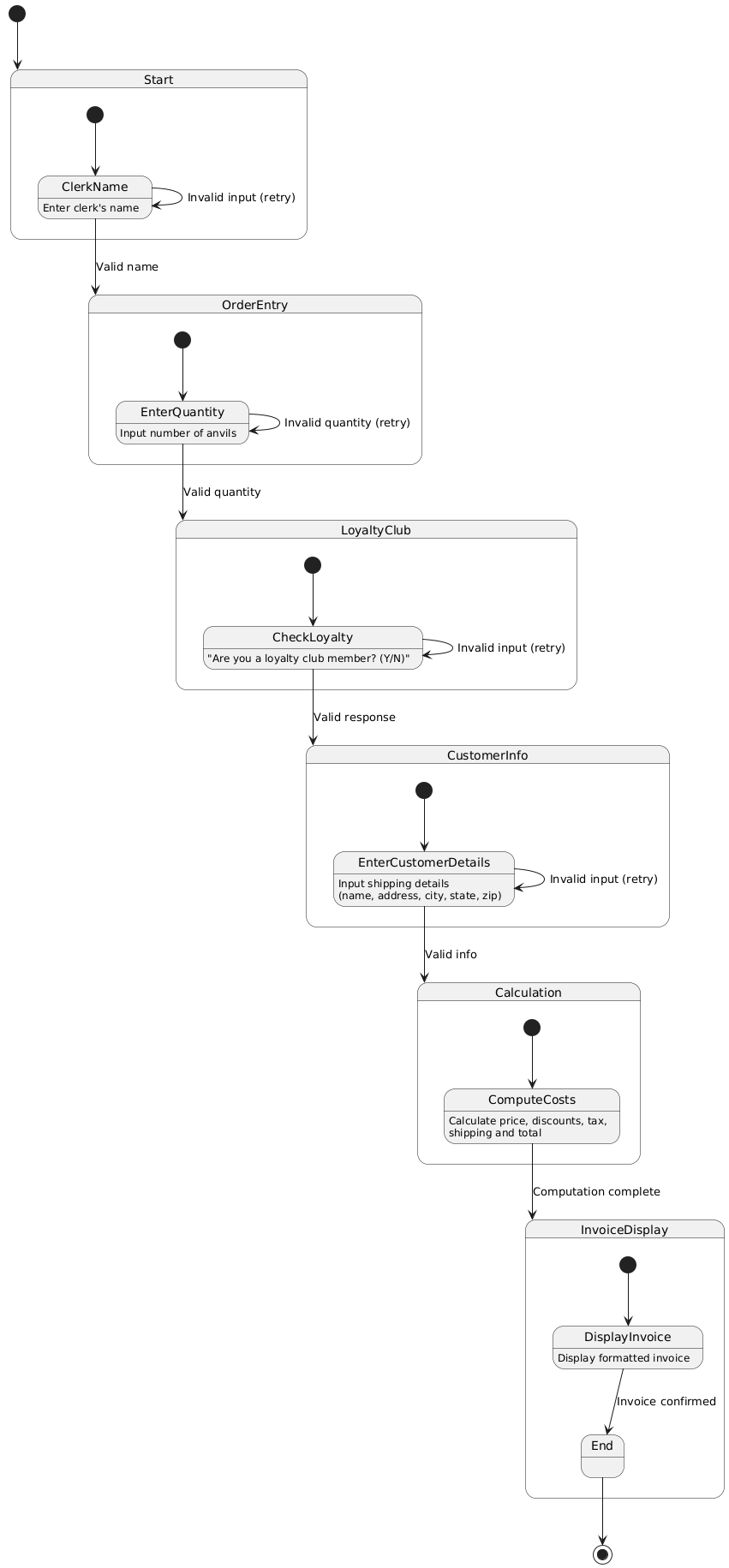
**5. State Transition Diagram**

**States:**

1. **Start** → Clerk enters name
2. **Order Entry** → Clerk inputs number of anvils
3. **Loyalty Club entry** → Clerk inputs if customer is a Futility Club member
4. **Customer Info** → Clerk enters shipping details
5. **Calculation** → System computes price, subtotal, loyalty discount, taxable amount, tax, shipping discount, and total
6. **Invoice Display** → Invoice is shown to the user
7. **End**

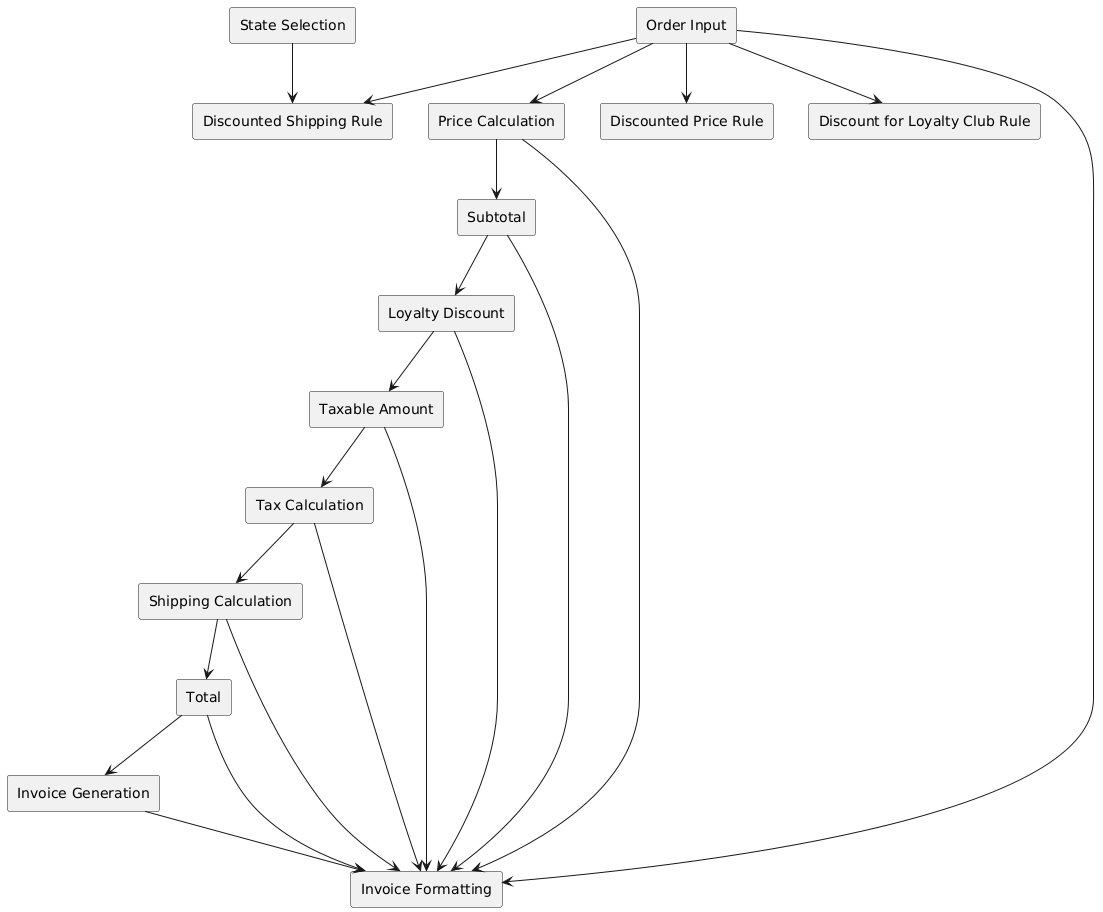
**Transitions:**

* **Valid input → next state**
* **Invalid input → retry state**
* **Final invoice display → end state**



**6. Dependency Graphs**

Illustrating dependency between different modules:

1. **Order Input** ⟶ **Price Calculation** ⟶**Subtotal**  → **Loyalty Discount** → **Taxable Amount** → **Tax Calculation** ⟶ **Shipping Calculation** → **Total**⟶ **Invoice Generation**
2. Discounted Shipping Rule **depends** on **Order Input** and **State Selection**.
3. Discounted Price Rule depends on Order Input.
4. Discount for Loyalty Club Rule depends on Order Input.
5. **Invoice Formatting** depends on all previous calculations. 

Technical Requirements

## Non-Functional Requirements

**1. Performance**

* **Response Time:** User input and calculations should be completed within **3 seconds**.
* **Throughput:** Support **100 orders per minute** without lag.

**2. Usability**

* **Ease of Use:** Guided order flow; completion within **2 minutes**.
* **Formatting:** Console output should be **readable, aligned, and intuitive**.
* **Learnability:** New users should operate efficiently within **15 minutes** of training.

**3. Availability & Reliability**

* **Uptime:** **99% availability** during business hours.
* **Error Handling:** Clear prompts for invalid inputs, ensuring **data integrity and fault tolerance**.

**4. Scalability**

* Support **10,000 orders per day** without performance issues.
* **Future-ready** for database integration and multi-order processing.

**5. Security**

* **Data Privacy:** No persistent storage of personal data without encryption.
* **Input Validation:** Prevent unexpected inputs and **injection attacks**.

**6. Maintainability**

* **Code Structure:** Modular, well-commented, and maintainable.
* **Error Logging:** Generate logs for troubleshooting.

**7. Portability**

* Compatible with **Windows, macOS, and Linux** with minimal dependencies.

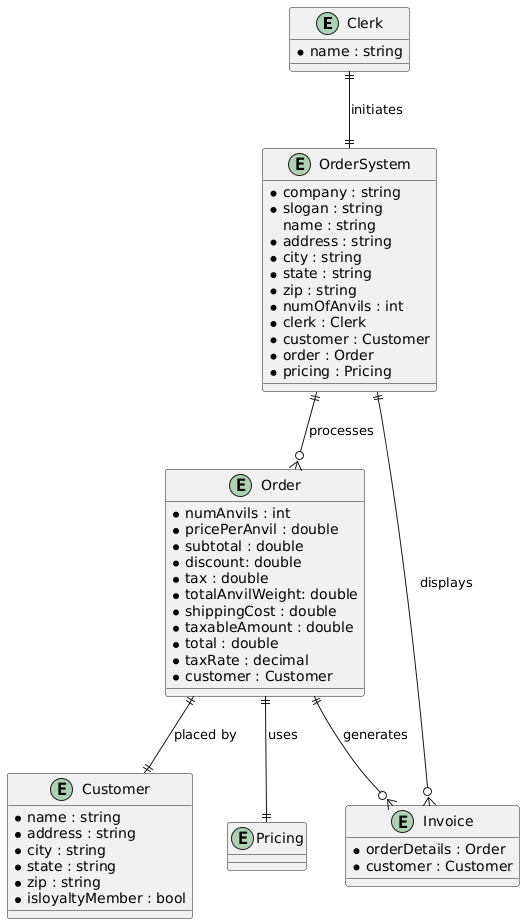
**8. Compliance**

* **Tax Compliance:** Enforce **9.5% tax** calculation.
* **Financial Accuracy:** Dollar amounts formatted correctly with **two decimal places**.

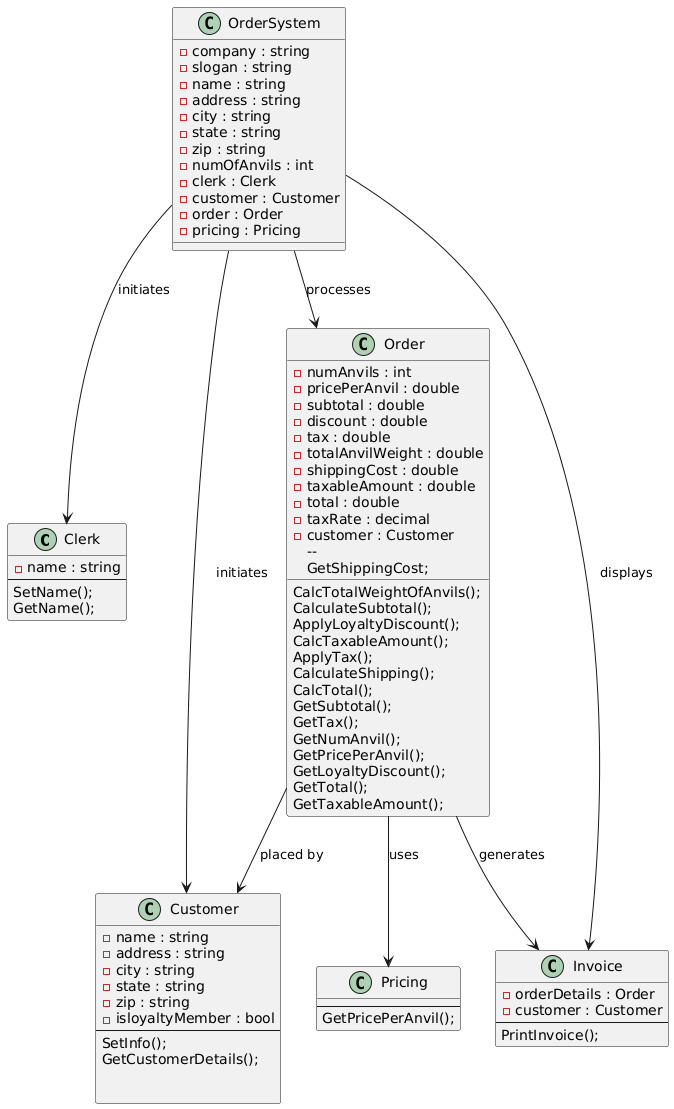
**9. Testing & Completion Criteria**

* **Unit Test Coverage:** **90%+** for calculations and input handling.
* **System Testing:** All functional and non-functional tests must pass.
* **Load Testing:** Simulate **500 concurrent orders** for performance validation

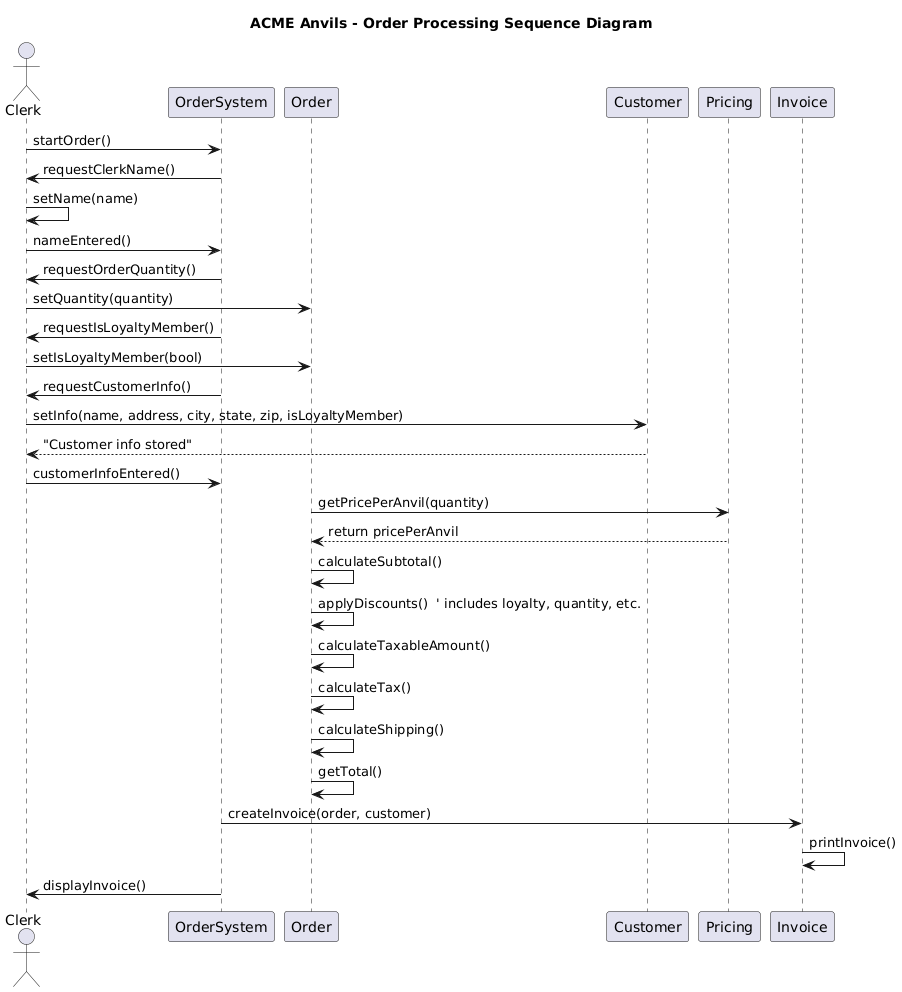
## Entity Relationship Diagram



## Class Diagrams



## Sequence Diagram



## Class Responsibility Collaboration

|  |  |
| --- | --- |
| **Clerk** | |
| Takes order from customer.  Inputs customer details.  Displays invoice to the customer. | OrderSystem (to start order processing)  Order (to store order details)  Customer (to store customer details) |

|  |  |
| --- | --- |
| **Customer** | |
| Stores shipping details.  Stores loyalty member status | Order (order is associated with a customer)  Invoice |

|  |  |
| --- | --- |
| **Order** | |
| Stores order details (number of anvils, shipping, loyalty discount, subtotal, taxable amount, tax, total).  Calculates shipping, loyalty discount, subtotal, taxable amount, tax, total.  Determines shipping cost. | Pricing (for per-anvil pricing).  Customer (for shipping address).  Invoice (to generate the invoice). |

|  |  |
| --- | --- |
| **Pricing** | |
| Provides per-anvil price based on quantity & discount. | Order (to calculate costs). |

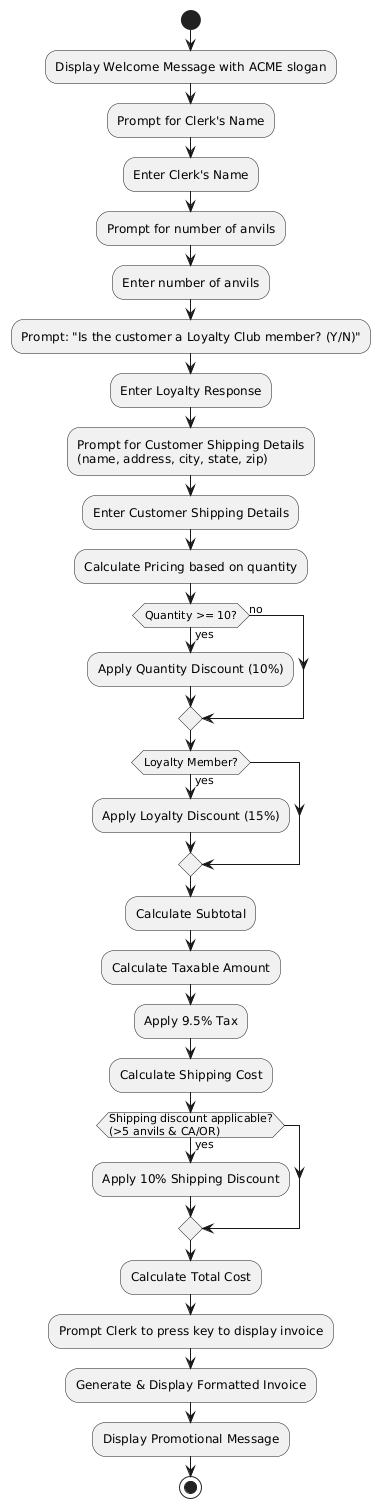
|  |  |
| --- | --- |
| **Invoice** | |
| Generates a formatted invoice with order details.  Displays invoice to the user. | Order (fetches order details).  Customer |

|  |  |
| --- | --- |
| **OrderSystem** | |
| Manages the overall flow of order processing.  Calls appropriate methods for calculations and invoice generation. | Clerk (user interaction).  Order (handles order details).  Invoice (for displaying final order summary).  Customer (initiates) |

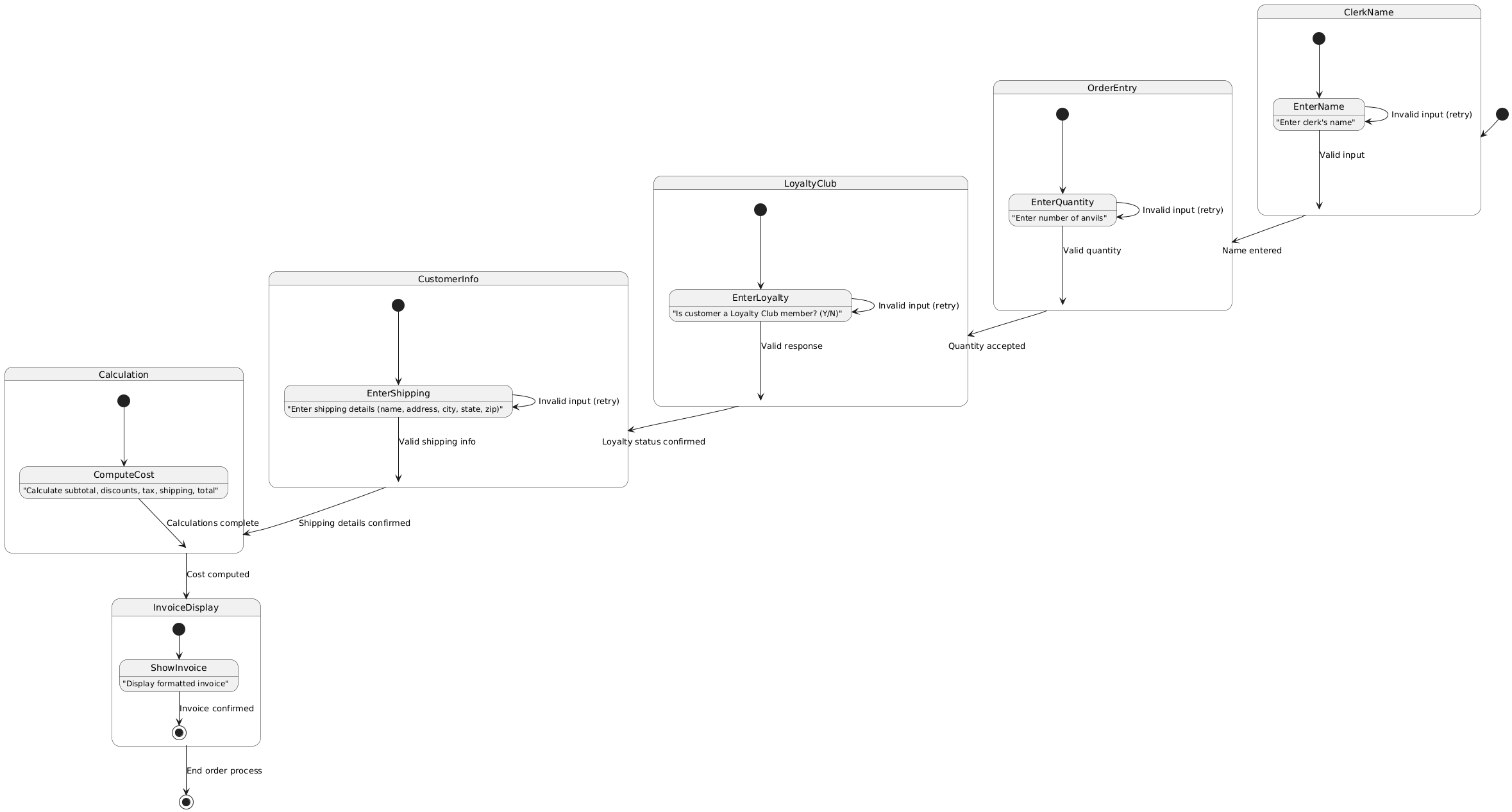
## Task Object Responsibility

|  |  |  |
| --- | --- | --- |
| **Task** | **Object** | **Responsibility** |
| Start order-taking process | OrderSystem | Initializes the order flow, prompting the clerk to begin. |
| Capture clerk’s name | Clerk | Inputs the clerk’s name so it can appear on the invoice. |
| Prompt for number of anvils | OrderSystem | Displays input prompt for the quantity of anvils the customer wants to order. |
| Capture customer details | Customer | Stores the customer’s name, address, city, state, zip, and loyalty club status. |
| Determine price per anvil | Pricing | Supplies the base per-anvil cost based on the quantity tier (1–9, 10–19, 20+). |
| Apply quantity discount | Order | Applies a 10% discount if the customer orders 10+ anvils (on top of tiered pricing). |
| Apply loyalty discount | Order | Applies a 15% discount if the customer is a loyalty club member. |
| Check for shipping discount | Pricing | Determines if the shipping cost should be discounted 10% for 6+ anvils shipped to CA or OR. |
| Calculate subtotal | Order | Multiplies the number of anvils by the (possibly discounted) per-anvil price. |
| Calculate taxable amount | Order | Subtotal minus any loyalty discount, setting the amount on which tax is calculated. |
| Calculate tax | Order | Applies a 9.5% tax rate to the taxable amount. |
| Calculate shipping | Order | Uses base shipping ($5/lbs × 50 lbs per anvil), then checks for shipping discount if applicable. |
| Calculate total | Order | Adds subtotal, tax, and shipping costs to get the final total. |
| Generate invoice | Invoice | Compiles all order details (customer info, discounts, shipping, tax, total) into a formatted invoice. |
| Display invoice to user | Invoice | Shows the final invoice details (subtotal, discounts, tax, shipping, total) to the clerk. |
| Confirm order completion | Clerk | Finalizes the order after reviewing the invoice with the customer. |
| End order session | OrderSystem | Terminates the process after the invoice is displayed, allowing the clerk to restart for the next order. |

## Activity Flow



## State Diagrams



## **Test Case 1**

**Test Case ID**: TC-UC1-001  
 **Use Case**: UC1 – Placing a Standard Order  
 **Actor**: Clerk  
 **Traceability**: FS-1, US6

### **Objective**

Verify that the system allows a clerk to place a standard order (e.g., 1–9 anvils), collect shipping info, apply correct pricing, and display the final invoice.

### **Preconditions**

* The system is installed and running.
* Clerk is ready to take an order.

### **Main Sequence**

1. **System Launch**: Clerk starts the ACME Anvil Order Taking System.
2. **Clerk Name**: System displays a welcome message and prompts for the clerk’s name; clerk enters name.
3. **Anvil Quantity**: System prompts for the number of anvils; clerk enters a valid number (e.g., 3).
4. **Loyalty Prompt**: System asks if the customer is a loyalty club member; clerk enters “N”.
5. **Shipping Details**: System requests the customer’s shipping info (name, address, city, state, zip); clerk enters valid data.
6. **Calculation**: System calculates total cost (subtotal, tax, shipping).
7. **Invoice Prompt**: System prompts clerk to press a key to display the invoice.
8. **Invoice Display**: System shows the invoice with all order details (subtotal, tax, shipping, total).

### **Alternative Sequence**

* **A1**: If the clerk enters an invalid anvil quantity (e.g., negative, non-numeric), the system may display an error and re-prompt. (Ties to UC5 if validation is extended.)

### **Postconditions**

* The final invoice is displayed with the correct calculations.
* The clerk is prompted to restart a new order if needed.

## **Test Case 2**

**Test Case ID**: TC-UC2-001  
 **Use Case**: UC2 – Ordering Greater Than 5 Anvils in CA or OR (Discounted Shipping)  
 **Actor**: Clerk  
 **Traceability**: FS-2, US5

### **Objective**

Verify that the system applies a 10% shipping discount when more than 5 anvils are shipped to California or Oregon.

### **Preconditions**

* The system is launched.
* Clerk is ready to take an order.

### **Main Sequence**

1. **Clerk Name**: Clerk enters their name when prompted.
2. **Anvil Quantity**: Clerk enters a quantity greater than 5 (e.g., 6).
3. **Loyalty Prompt**: System asks if the customer is a loyalty club member; clerk enters “N”.
4. **Shipping Details**: Clerk provides shipping info, ensuring **state = CA** or **OR**.
5. **Calculation**: System calculates shipping, applying a 10% discount due to quantity > 5 and CA/OR location.
6. **Invoice Prompt & Display**: Clerk presses a key, invoice is shown with discounted shipping.

### **Alternative Sequence**

* **A1**: If the clerk enters a different state or fewer than 6 anvils, the shipping discount is not applied (falls back to standard shipping).

### **Postconditions**

* The invoice reflects the discounted shipping cost.
* The order total is calculated correctly with tax, subtotal, and shipping.

## **Test Case 3**

**Test Case ID**: TC-UC3-001  
 **Use Case**: UC3 – Ordering 10+ Anvils for Discounted Pricing  
 **Actor**: Clerk  
 **Traceability**: FS-3, US3

### **Objective**

Verify that the system applies a 10% discount to the anvil unit price once the order quantity is 10 or more (leading to $63.00 per anvil for 10–19).

### **Preconditions**

* The system is running.
* Clerk is prepared to enter an order.

### **Main Sequence**

1. **Clerk Name**: Clerk enters their name.
2. **Anvil Quantity**: Clerk enters a quantity of 10 anvils.
3. **Loyalty Prompt**: Clerk indicates “N” for loyalty membership.
4. **Shipping Details**: Clerk provides valid shipping info.
5. **Price Calculation**: System calculates the per-anvil price at $63.00 (10% off $70.00).
6. **Invoice Display**: Clerk presses a key, invoice shows discounted price per anvil.

### **Alternative Sequence**

* **A1**: If clerk enters 9 or fewer anvils, no quantity discount is applied.
* **A2**: If clerk enters 20 or more anvils, the bulk discount from UC4 applies instead.

### **Postconditions**

* The invoice displays the correct discounted anvil price ($63.00).
* Subtotal, tax, shipping, and final total reflect the new price tier.

## **Test Case 4**

**Test Case ID**: TC-UC4-001  
 **Use Case**: UC4 – Ordering 20+ Anvils for Bulk Discount  
 **Actor**: Clerk  
 **Traceability**: FS-4, US3

### **Objective**

Ensure that when the order quantity is 20 or more, the system applies the bulk discount of 10% off the $68.25 tier, resulting in $61.425 per anvil.

### **Preconditions**

* System is operational.
* Clerk is ready to place a large order.

### **Main Sequence**

1. **Clerk Name**: Clerk enters name at the prompt.
2. **Anvil Quantity**: Clerk enters 20 (or more) anvils.
3. **Loyalty Prompt**: Clerk enters “N” for loyalty membership.
4. **Shipping Details**: Clerk enters valid shipping data.
5. **Bulk Calculation**: System calculates the per-anvil price at $61.425.
6. **Invoice Display**: Clerk presses a key to display the invoice with the bulk discount.

### **Alternative Sequence**

* **A1**: If the clerk enters fewer than 20 anvils, the system applies a different tier (UC3 or standard).
* **A2**: If loyalty membership is “Y”, an additional 15% discount might apply on top of the discounted tier price (per UC6).

### **Postconditions**

* Invoice shows the bulk discount pricing.
* Subtotal, tax, shipping, and total are correctly calculated.

## **Test Case 5**

**Test Case ID**: TC-UC6-001  
 **Use Case**: UC6 – Customer is a Futility Club Member (Apply Discount)  
 **Actor**: Clerk  
 **Traceability**: FS-6, US4

### **Objective**

Validate that the system applies a 15% loyalty discount for Futility Club members and shows it clearly on the invoice.

### **Preconditions**

* System is running.
* Clerk is ready to take an order.

### **Main Sequence**

1. **Clerk Name**: Clerk enters name.
2. **Anvil Quantity**: Clerk enters a valid quantity (e.g., 5).
3. **Loyalty Prompt**: System asks about loyalty membership; clerk enters “Y”.
4. **Shipping Details**: Clerk enters valid shipping data.
5. **Discount Calculation**: System applies a 15% discount to the subtotal.
6. **Invoice Display**: Clerk presses a key, invoice shows “Less 15% Loyalty Club” line and final total.

### **Alternative Sequence**

* **A1**: If clerk enters “N” for loyalty membership, no loyalty discount is applied (reverts to standard or other discount logic if applicable).

### **Postconditions**

* Invoice reflects the loyalty discount line item.
* The final total is adjusted by 15% discount on the subtotal (and any other applicable discounts).

Executable Tests

Customer Tests

|  |  |
| --- | --- |
| **Test Case: Sets\_CustomerInfo\_WhenSet** | |
| **ID** | **TST-Cust-001** |
| **Unit Tested** | Sets\_CustomerInfo\_WhenSet() |
| **Preconditions** | 1. Visual Studio MSTest environment ready 2. Customer class compiled without errors |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | - Name: "Toco" - Street: "123 elm st" - City: "Bellevue" - State: "WA" - Zip: "98005" - LoyaltyClub: false |
| **Steps** | 1. Instantiate a new Customer object. 2. Call customer.SetInfo(...) with the data above. 3. Call customer.GetCustomerDetails(). |
| **Expected Results** | 1. Returned string should match: "\tToco\n\t123 elm st\n\tBellevue\n\tWA\n\t98005" 2. Test framework Assert.AreEqual passes with the expected value. |
| **Actual Results** | The expected and actual are equal |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Low |
| **Comments** | Verifies basic SetInfo() → GetCustomerDetails() chain. |

|  |  |
| --- | --- |
| **Test Case: Get\_CustomerDetails\_FormatsOutputCorrectly** | |
| **ID** | **TST-Cust-002** |
| **Unit Tested** | Get\_CustomerDetails\_FormatsOutputCorrectly() |
| **Preconditions** | 1. MSTest environment ready 2. Customer class compiled successfully |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | - Name: "Bob Doe" - Street: "123 Elm St" - City: "Seattle" - State: "NY" - Zip: "12345" - LoyaltyClub: false |
| **Steps** | 1. Instantiate a new Customer. 2. Call customer.SetInfo("Bob Doe", "123 Elm St", "Seattle", "NY", "12345", false).  3. Invoke string result = customer.GetCustomerDetails().  4. Check if result contains each of the test data elements. |
| **Expected Results** | 1. result must contain "Bob Doe", "123 Elm St", "Seattle", "NY", and "12345".  2. Test asserts (e.g., Assert.IsTrue) all pass. |
| **Actual Results** | All asserts are true with the string containing: "Bob Doe", "123 Elm St", "Seattle", "NY", and "12345". |
| **Pass/Fail** | Pass |
| **Priority** | Low |
| **Severity** | Low |
| **Comments** | Ensures correct string formatting and presence of all fields. |

|  |  |
| --- | --- |
| **Test Case: Returns\_FormattedString\_WhenValuesAreEmpty** | |
| **ID** | **TST-Cust-003** |
| **Unit Tested** | Returns\_FormattedString\_WhenValuesAreEmpty() |
| **Preconditions** | 1. MSTest environment 2. Customer class compiled |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | - Name: "" (empty) - Street: "" - City: "" - State: "" - Zip: "" - LoyaltyClub: false |
| **Steps** | 1. Instantiate Customer. 2. Call customer.SetInfo("", "", "", "", "", false).  3. Get result: customer.GetCustomerDetails(). |
| **Expected Results** | 1. Expected string: "\t\n\t\n\t\n\t\n\t" (tabs and newlines only). 2. Assert.AreEqual(expected, result) passes. |
| **Actual Results** | Assert of actual and expect are equal passed |
| **Pass/Fail** | Pass |
| **Priority** | Medium |
| **Severity** | Low |
| **Comments** | Validates that no null reference or missing field errors occur with empty values. |

|  |  |
| --- | --- |
| **Test Case: Set\_AccuratlyUpdatesExistingValues\_WhenNewValuesAreIntroduced** | |
| **ID** | **TST-Cust-004** |
| **Unit Tested** | Set\_AccuratlyUpdatesExistingValues\_WhenNewValuesAreIntroduced() |
| **Preconditions** | 1. MSTest environment 2. Customer class compiled |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | - Initial: "Initial Name", "Initial Street", "Initial City", "IC", "00000", false - Updated: "Updated Name", "Updated Street", "Updated City", "UC", "11111", false |
| **Steps** | 1. Instantiate Customer. 2. customer.SetInfo(initialValues...). 3. customer.SetInfo(updatedValues...). 4. result = customer.GetCustomerDetails(). |
| **Expected Results** | result = "\tUpdated Name\n\tUpdated Street\n\tUpdated City\n\tUC\n\t11111". |
| **Actual Results** | Assert of actual and expect are equal passed |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | Ensures second call to SetInfo() overrides previous values. |

|  |  |
| --- | --- |
| **Test Case: ReturnDetails\_ContainingAllRequiredElements\_WhenInfoIsSet** | |
| **ID** | **TST-Cust-005** |
| **Unit Tested** | ReturnDetails\_ContainingAllRequiredElements\_WhenInfoIsSet() |
| **Preconditions** | MSTest environment, Customer class compiled |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | "Test Customer", "Test Address", "Test City", "TC", "12345", false |
| **Steps** | 1. Create Customer object. 2. SetInfo("Test Customer", "Test Address", "Test City", "TC", "12345", false).  3. details = customer.GetCustomerDetails().  4. Verify it contains all substrings: "Test Customer", "Test Address", "Test City", "TC", "12345", "\t", and "\n". |
| **Expected Results** | All required substrings are present in details; test passes. |
| **Actual Results** | All asserts are true with the string containing: "Test Customer", "Test Address", "Test City", "TC", "12345", \n, \t |
| **Pass/Fail** | Pass |
| **Priority** | Medium |
| **Severity** | Low |
| **Comments** | Confirms final detail string includes tabs/newlines plus all fields. |

|  |  |
| --- | --- |
| **Test Case: ReturnsFalse\_WhenLoyaltyStatusIsFalse** | |
| **ID** | **TST-Cust-006** |
| **Unit Tested** | ReturnsFalse\_WhenLoyaltyStatusIsFalse(...) |
| **Preconditions** | MSTest environment, Customer class compiled |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | DataRow 1: "Test Customer", "Test Address", "Test City", "TC", "12345", false DataRow 2: "Test Customer2", "Test Address2", "Test City", "TC", "12345", false DataRow 3: "Test Customer3", "Test Address3", "Test City", "TC", "12345", false |
| **Steps** | 1. For each row, instantiate Customer. 2. Call SetInfo(...) with that row’s data (loyaltyClub = false). 3. Invoke bool result = customer.GetLoyaltyStatus(). |
| **Expected Results** | result is always false for each row (Assert.IsFalse). |
| **Actual Results** | All 3 datarows inputs result in a isFalse result |
| **Pass/Fail** | Pass |
| **Priority** | Medium |
| **Severity** | Medium |
| **Comments** | Tests multiple data sets to ensure loyalty status returns false. |

|  |  |
| --- | --- |
| **Test Case: ReturnsTrue\_WhenLoyaltyStatusIsTrue** | |
| **ID** | **TST-Cust-007** |
| **Unit Tested** | ReturnsTrue\_WhenLoyaltyStatusIsTrue(...) |
| **Preconditions** | MSTest environment, Customer class compiled |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | DataRow 1: "Test Customer", "Test Address", "Test City", "TC", "12345", true DataRow 2: "Test Customer2", "Test Address2", "Test City", "TC", "12345", true DataRow 3: "Test Customer3", "Test Address3", "Test City", "TC", "12345", true |
| **Steps** | 1. For each data row, create Customer. 2. Call SetInfo(...) with loyaltyClub = true. 3. bool result = customer.GetLoyaltyStatus(). |
| **Expected Results** | result is true for each row (Assert.IsTrue). |
| **Actual Results** | All 3 datarows inputs result in a isTrue result |
| **Pass/Fail** | Pass |
| **Priority** | Medium |
| **Severity** | Medium |
| **Comments** | Verifies loyalty flag is accurately stored and returned. |

Pricing Tests

|  |  |
| --- | --- |
| Test Case: Return\_PricePerAnvil\_WithCorrectPrice | |
| **ID** | **TST-Price-001** |
| **Unit Tested** | Return\_PricePerAnvil\_WithCorrectPrice(int quantity, double expectedPrice) |
| **Preconditions** | 1. MSTest environment is set up.  2. Pricing class is compiled without errors. |
| **Environment** | .NET environment, MSTest framework, local development machine |
| **Test Data** | Sub-Case A: quantity = -1, expected = 0.0  Sub-Case B: quantity = 0, expected = 0.0  Sub-Case C: quantity = 1, expected = 88.50  Sub-Case D: quantity = 5, expected = 88.50  Sub-Case E: quantity = 9, expected = 88.50  Sub-Case F: quantity = 10, expected = 63.00  Sub-Case G: quantity = 15, expected = 63.00  Sub-Case H: quantity = 19, expected = 63.00  Sub-Case I: quantity = 20, expected = 61.425  Sub-Case J: quantity = 50, expected = 61.425  Sub-Case K: quantity = 100, expected = 61.425 |
| **Steps** | 1. Instantiate Pricing class if needed (already done via \_pricing). 2. For each sub-case (DataRow), call double actualPrice = \_pricing.GetPricePerAnvil(quantity). 3. Compare actualPrice to expectedPrice within a small delta (0.001). |
| **Expected Results** | For each sub-case, Assert.AreEqual(expectedPrice, actualPrice, 0.001) passes.  Sub-Case A: -1 → 0.0  Sub-Case B: 0 → 0.0  Sub-Case C: 1 → 88.50  Sub-Case D: 5 → 88.50  Sub-Case E: 9 → 88.50  Sub-Case F: 10 → 63.00  Sub-Case G: 15 → 63.00  Sub-Case H: 19 → 63.00  Sub-Case I: 20 → 61.425  Sub-Case J: 50 → 61.425  Sub-Case K: 100 → 61.425 |
| **Actual Results** | Sub-Case A: -1 → 0.0  Sub-Case B: 0 → 0.0  Sub-Case C: 1 → 88.50  Sub-Case D: 5 → 88.50  Sub-Case E: 9 → 88.50  Sub-Case F: 10 → 63.00  Sub-Case G: 15 → 63.00  Sub-Case H: 19 → 63.00  Sub-Case I: 20 → 61.425  Sub-Case J: 50 → 61.425  Sub-Case K: 100 → 61.425 |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | Confirms the pricing tiers (and fallback to 0 for invalid/negative quantity).  Pricing discount per quantity needs further confirmation from client. |

Order Tests

|  |  |
| --- | --- |
| **Test Case: Returns\_CorrectCalculation\_NoLoyaltyIncorrectValue** | |
| **ID** | **TST-Order-001** |
| **Unit Tested** | Returns\_CorrectCalculation\_NoLoyaltyIncorrectValue(int quantity, double pricePerAnvil) |
| **Preconditions** | 1. MSTest environment set up 2. Customer and Order classes compiled successfully |
| **Environment** | .NET environment, MSTest framework, local development machine |
| **Test Data** | Sub-Cases (loyalty = false): • Quantity = -5, Price per anvil = 88.50 • Quantity = -1, Price per anvil = 88.50 • Quantity = 0, Price per anvil = 88.50 |
| **Steps** | 1. Create a shipping cost constant: OneAnvilShippingCost = 5.00 × 50. 2. Instantiate a Customer and call SetInfo("Toco", "123 elm st", "Bellevue", "NY", "98005", false). 3. For each sub-case, create an Order using the test quantity and price. 4. Calculate the expected values:  • Subtotal = pricePerAnvil × quantity  • Tax = Subtotal × 0.095  • Shipping = OneAnvilShippingCost × quantity  • Total = Subtotal + Tax + Shipping 5. Verify that the order’s GetSubtotal, GetTax, GetShippingCost, and GetTotal methods return the expected values within a delta of 0.001. |
| **Expected Results** | For each sub-case:  • Order.GetSubtotal() equals (pricePerAnvil × quantity)  • Order.GetTax() equals (pricePerAnvil × quantity × 0.095)  • Order.GetShippingCost() equals (OneAnvilShippingCost × quantity)  • Order.GetTotal() equals the sum of the above values. |
| **Actual Results** | For all 3 data row entries: SubTotal actual and expected are equal  Tax actual and expected are equal  Shipping actual and expected are equal  Total actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | Verifies that negative or zero quantity values return a calculation based on the input (which may result in negative or zero amounts) when loyalty is not applied. |

|  |  |
| --- | --- |
| **Test Case: Returns\_CorrectCalculation\_NoLoyaltySmallOrderQuantity** | |
| **ID** | **TST-Order-002** |
| **Unit Tested** | Returns\_CorrectCalculation\_NoLoyaltySmallOrderQuantity1(int quantity, double pricePerAnvil) |
| **Preconditions** | MSTest environment is configured; Customer and Order classes compile without error. |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | Sub-Cases (loyalty = false): • Quantity = 1, Price = 88.50 • Quantity = 2, Price = 88.50 • Quantity = 5, Price = 88.50 • Quantity = 9, Price = 88.50 |
| **Steps** | 1. Define OneAnvilShippingCost = 5.00 × 50. 2. Instantiate a Customer with SetInfo("Toco", "123 elm st", "Bellevue", "NY", "98005", false).  3. For each sub-case, create an Order with the given quantity and price. 4. Calculate expected values:  • Subtotal = pricePerAnvil × quantity  • Tax = Subtotal × 0.095  • Shipping = OneAnvilShippingCost × quantity  • Total = Subtotal + Tax + Shipping 5. Assert that the order’s calculated values match the expected ones within Delta. |
| **Expected Results** | All assertions pass with:  • Expected Subtotal = pricePerAnvil × quantity  • Expected Tax = (pricePerAnvil × quantity) × 0.095  • Expected Shipping = OneAnvilShippingCost × quantity  • Expected Total = Subtotal + Tax + Shipping |
| **Actual Results** | For all 4 data row entries: SubTotal actual and expected are equal  Tax actual and expected are equal  Shipping actual and expected are equal  Total actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | Confirms correct calculation for small order quantities when the customer is not a loyalty member. |

|  |  |
| --- | --- |
| **Test Case: Returns\_CorrectCalculation\_YesLoyaltySmallOrderQuantity** | |
| **ID** | **TST-Order-003** |
| **Unit Tested** | Returns\_CorrectCalculation\_YesLoyaltySmallOrderQuantity(int quantity, double pricePerAnvil) |
| **Preconditions** | MSTest environment ready; Customer and Order classes compiled. |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | Sub-Cases (loyalty = true): • Quantity = 1, Price = 88.50 • Quantity = 2, Price = 88.50 • Quantity = 5, Price = 88.50 • Quantity = 9, Price = 88.50 |
| **Steps** | 1. Set OneAnvilShippingCost = 5.00 × 50. 2. Create a Customer with SetInfo("Toco", "123 elm st", "Bellevue", "NY", "98005", true) (loyalty = true). 3. For each sub-case, instantiate an Order with the given quantity and price. 4. Compute expected values:  • Subtotal = pricePerAnvil × quantity  • Loyalty Discount = Subtotal × 0.15  • Taxable Amount = Subtotal − Loyalty Discount  • Tax = Taxable Amount × 0.095  • Shipping = OneAnvilShippingCost × quantity  • Total = Taxable Amount + Tax + Shipping 5. Verify each calculated value using assertions within Delta. |
| **Expected Results** | Each assertion passes:  • Order.GetSubtotal() equals pricePerAnvil × quantity  • Order.GetLoyaltyDiscount() equals Subtotal × 0.15  • Order.GetTaxableAmount() equals Subtotal − (Subtotal × 0.15)  • Order.GetTax() equals (Taxable Amount) × 0.095  • Order.GetShippingCost() equals OneAnvilShippingCost × quantity  • Order.GetTotal() equals Taxable Amount + Tax + Shipping |
| **Actual Results** | For all 4 data row entries: Discount actual and expected are equal  Taxable actual and expected are equal  SubTotal actual and expected are equal  Tax actual and expected are equal  Shipping actual and expected are equal  Total actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | Ensures that loyalty discounts are correctly applied for small orders. |

|  |  |
| --- | --- |
| **Test Case: Returns\_CorrectCalculation\_NoLoyaltyMidOrderQuantity** | |
| **ID** | **TST-Order-004** |
| **Unit Tested** | Returns\_CorrectCalculation\_NoLoyaltyMidOrderQuantity10(int quantity, double pricePerAnvil) |
| **Preconditions** | MSTest environment ready; required classes compile successfully. |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | Sub-Cases (loyalty = false): • Quantities: 10, 11, 15, 18, 19 with Price = 63.00 |
| **Steps** | 1. Set OneAnvilShippingCost = 5.00 × 50. 2. Create a Customer with SetInfo("Toco", "123 elm st", "Bellevue", "NY", "98005", false). 3. For each sub-case, instantiate an Order with the specified quantity and price. 4. Calculate expected values:  • Subtotal = pricePerAnvil × quantity  • Tax = Subtotal × 0.095  • Shipping = OneAnvilShippingCost × quantity  • Total = Subtotal + Tax + Shipping 5. Assert that each returned value matches the expected calculations within Delta. |
| **Expected Results** | All calculated values (Subtotal, Tax, Shipping, Total) are correct as defined above. |
| **Actual Results** | For all 5 data row entries: SubTotal actual and expected are equal  Tax actual and expected are equal  Shipping actual and expected are equal  Total actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | Tests the mid-range order calculations without loyalty discount. |

|  |  |
| --- | --- |
| **Test Case: Returns\_CorrectCalculation\_YesLoyaltyMidOrderQuantity** | |
| **ID** | **TST-Order-005** |
| **Unit Tested** | Returns\_CorrectCalculation\_YesLoyaltyMidOrderQuantity10(int quantity, double pricePerAnvil) |
| **Preconditions** | MSTest environment is set up; necessary classes compile. |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | Sub-Cases (loyalty = true): • Quantities: 10, 11, 15, 18, 19 with Price = 63.00 |
| **Steps** | 1. Define OneAnvilShippingCost = 5.00 × 50. 2. Instantiate a Customer with loyalty set to true using SetInfo("Toco", "123 elm st", "Bellevue", "NY", "98005", true). 3. For each sub-case, create an Order with the provided quantity and price. 4. Expected calculations:  • Subtotal = pricePerAnvil × quantity  • Loyalty Discount = Subtotal × 0.15  • Taxable Amount = Subtotal − Loyalty Discount  • Tax = Taxable Amount × 0.095  • Shipping = OneAnvilShippingCost × quantity  • Total = Taxable Amount + Tax + Shipping 5. Verify each value using assertions within Delta. |
| **Expected Results** | The order’s values match the above calculations. |
| **Actual Results** | For all 5 data row entries: Discount actual and expected are equal  Taxable actual and expected are equal  SubTotal actual and expected are equal  Tax actual and expected are equal  Shipping actual and expected are equal  Total actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | Checks correct application of both quantity pricing and loyalty discounts in mid-range orders. |

|  |  |
| --- | --- |
| **Test Case: Returns\_CorrectCalculation\_NoLoyaltyBigOrderQuantity** | |
| **ID** | **TST-Order-006** |
| **Unit Tested** | Returns\_CorrectCalculation\_NoLoyaltyBigOrderQuantity20(int quantity, double pricePerAnvil) |
| **Preconditions** | MSTest environment ready; Customer and Order classes available. |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | Sub-Cases (loyalty = false): • Quantities: 20, 21, 30 with Price = 61.425 |
| **Steps** | 1. Calculate OneAnvilShippingCost = 5.00 × 50. 2. Create a Customer with SetInfo("Toco", "123 elm st", "Bellevue", "NY", "98005", false). 3. For each sub-case, instantiate an Order with the given quantity and price. 4. Expected values:  • Subtotal = pricePerAnvil × quantity  • Tax = Subtotal × 0.095  • Shipping = OneAnvilShippingCost × quantity  • Total = Subtotal + Tax + Shipping 5. Assert that the returned values match the expected ones within Delta. |
| **Expected Results** | All assertions pass for Subtotal, Tax, Shipping, and Total. |
| **Actual Results** | For all 5 data row entries: SubTotal actual and expected are equal  Tax actual and expected are equal  Shipping actual and expected are equal  Total actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | Verifies correct bulk order calculations for non-loyalty customers. |

|  |  |
| --- | --- |
| **Test Case: Returns\_CorrectCalculation\_YesLoyaltyBigOrderQuantity** | |
| **ID** | **TST-Order-007** |
| **Unit Tested** | Returns\_CorrectCalculation\_YesLoyaltyBigOrderQuantity20(int quantity, double pricePerAnvil) |
| **Preconditions** | MSTest environment is active; classes compile without error. |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | Sub-Cases (loyalty = true): • Quantities: 20, 21, 30 with Price = 61.425 |
| **Steps** | 1. Set OneAnvilShippingCost = 5.00 × 50. 2. Create a Customer with loyalty = true using SetInfo("Toco", "123 elm st", "Bellevue", "NY", "98005", true). 3. For each sub-case, create an Order with the test data. 4. Expected calculations:  • Subtotal = pricePerAnvil × quantity  • Loyalty Discount = Subtotal × 0.15  • Taxable Amount = Subtotal − Loyalty Discount  • Tax = Taxable Amount × 0.095  • Shipping = OneAnvilShippingCost × quantity  • Total = Taxable Amount + Tax + Shipping 5. Validate using assertions within Delta. |
| **Expected Results** | Order values (Subtotal, Loyalty Discount, Taxable Amount, Tax, Shipping, Total) match expected calculations. |
| **Actual Results** | For all 5 data row entries: Discount actual and expected are equal  Taxable actual and expected are equal  SubTotal actual and expected are equal  Tax actual and expected are equal  Shipping actual and expected are equal  Total actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | Ensures that bulk orders for loyalty members are calculated correctly. |

|  |  |
| --- | --- |
| **Test Case: Returns\_CorrectCalculation\_NoLoyaltyDiscountShippingCA\_OR\_6orMoreAnvils** | |
| **ID** | **TST-Order-008** |
| **Unit Tested** | Returns\_CorrectCalculation\_NoLoyaltyDiscountShippingCA\_OR\_6orMoreAnvils(int quantity, double pricePerAnvil, string state) |
| **Preconditions** | MSTest environment ready; classes compiled. Note: Although a “state” parameter is provided in DataRow. |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | Sub-Cases (loyalty = false, intended for CA/OR shipping discount tests): • For CA: Quantities – 1, 2, 5 (no discount) and 6, 9, 10, 11, 15, 18, 19, 20, 21, 30 (discount scenario as per use case description) • For OR: Same as CA with respective DataRow values |
| **Steps** | 1. Define OneAnvilShippingCost = 5.00 × 50. 2. Create a Customer with SetInfo("Toco", "123 elm st", "Bellevue", state data, "98005", false).  3. For each DataRow, instantiate an Order using the given quantity and price. 4. Expected values:  • Subtotal = pricePerAnvil × quantity  • Tax = Subtotal × 0.095  • Shipping = OneAnvilShippingCost × quantity  • Total = Subtotal + Tax + Shipping 5. Verify each assertion within Delta. |
| **Expected Results** | For each sub-case, the order’s Subtotal, Tax, Shipping, and Total match the above calculations. |
| **Actual Results** | For all 26 data row entries: SubTotal actual and expected are equal  Tax actual and expected are equal  Shipping actual and expected are equal  Total actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | Medium |
| **Severity** | Medium |
| **Comments** | Ensures Shipping Discount is calculated with no loyalty discount for CA and OR states. |

|  |  |
| --- | --- |
| **Test Case: Returns\_CorrectCalculation\_YesLoyaltyDiscountShippingCA\_OR\_6orMoreAnvils** | |
| **ID** | **TST-Order-009** |
| **Unit Tested** | Returns\_CorrectCalculation\_YesLoyaltyDiscountShippingCA\_OR\_6orMoreAnvils(int quantity, double pricePerAnvil, string state) |
| **Preconditions** | MSTest environment ready; required classes compile. Same note as above regarding the state value. |
| **Environment** | .NET environment, MSTest framework, local dev machine |
| **Test Data** | Sub-Cases (loyalty = true, intended for CA/OR discount testing): • For CA and OR: DataRows with various quantities (from <6 up to 10+ anvils) using the corresponding price per anvil values as defined (e.g., 88.50 for small orders, 63.00 for mid-range, 61.425 for large orders) |
| **Steps** | 1. Set OneAnvilShippingCost = 5.00 × 50. 2. Instantiate a Customer with loyalty enabled via SetInfo("Toco", "123 elm st", "Bellevue", state date, "98005", true). 3. For each DataRow, create an Order using the test quantity and price. 4. Expected calculations:  • Subtotal = pricePerAnvil × quantity  • Loyalty Discount = Subtotal × 0.15  • Taxable Amount = Subtotal − Loyalty Discount  • Tax = Taxable Amount × 0.095  • Shipping = OneAnvilShippingCost × quantity  • Total = Taxable Amount + Tax + Shipping 5. Assert that each returned value is within Delta of the expected value. |
| **Expected Results** | Each assertion passes with values matching the above calculations. |
| **Actual Results** | For all 26 data row entries: Discount actual and expected are equal  Taxable actual and expected are equal  SubTotal actual and expected are equal  Tax actual and expected are equal  Shipping actual and expected are equal  Total actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | Medium |
| **Severity** | Medium |
| **Comments** | Verifies combined loyalty discount and shipping/quantity discount calculations for orders that would qualify under CA/OR criteria. |

Invoice Tests

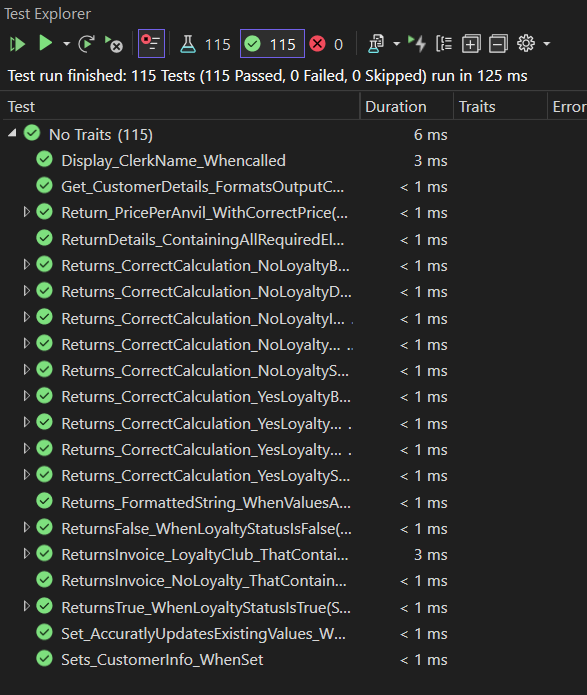
|  |  |
| --- | --- |
| **Test Case 1: ReturnsInvoice\_NoLoyalty\_ThatContainsRequiredElements** | |
| **ID** | **TST-INV-001** |
| **Unit Tested** | ReturnsInvoice\_NoLoyalty\_ThatContainsRequiredElements() |
| **Preconditions** | 1. MSTest environment is configured. 2. The Customer, Order, and Invoice classes compile without errors. 3. Console output can be redirected. |
| **Environment** | .NET environment, MSTest framework, local development machine |
| **Test Data** | - Customer details:   • Name: "Test Customer"   • Street: "123 Test St"   • City: "Test City"   • State: "TS"   • Zip: "12345"   • Loyalty: false   - Order details:   • Quantity: 2   • Price per anvil: 88.50 |
| **Steps** | 1. Call helper method TestCustomerLoyaltyFalse() to obtain a customer with loyalty set to false. 2. Create an Order by calling TestOrder(2, 88.50, customer).  3. Instantiate an Invoice using the order and customer. 4. Redirect the console output to a StringWriter to capture the printed invoice. 5. Call invoice.PrintInvoice().  6. Capture the output string. 7. Verify that the output contains each required element:  • "ACME Anvils Corporation"  • "Customer Invoice"  • "SHIP TO:"  • "Test Customer", "123 Test St", "Test City", "TS", "12345"  • "Quantity ordered:"  • "Cost per anvil:"  • "Subtotal:"  • "Sales Tax:"  • "Shipping:"  • "TOTAL:" |
| **Expected Results** | The invoice output must include all the above substrings are contained in the output so that the invoice displays the required header, customer address, order details, cost summary, and total. |
| **Actual Results** | All required substrings are contained in all datarow outputs |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | This test confirms that, for a non–loyalty customer, the invoice prints the company header, customer shipping information, and cost summary without any loyalty discount details. |

|  |  |
| --- | --- |
| **Test Case 2: ReturnsInvoice\_LoyaltyClub\_ThatContainsRequiredElements** | |
| **ID** | **TST-Inv-002** |
| **Unit Tested** | ReturnsInvoice\_LoyaltyClub\_ThatContainsRequiredElements(int qty, double price, string name, string street, string city, string state, string zip, bool loyaltyClub) |
| **Preconditions** | 1. MSTest environment is configured. 2. The Customer, Order, and Invoice classes compile without errors. 3. Console output redirection is supported. |
| **Environment** | .NET environment, MSTest framework, local development machine |
| **Test Data** | This test is executed using multiple DataRow sub-cases. For example:  • Sub-Case A: qty = 1, price = 88.50, name = "Test Customer", street = "123 Test St", city = "Test City", state = "TS", zip = "12345", loyalty = true  • Sub-Case B: qty = 2, price = 88.50, name = "Test Customer", street = "123 Test St", city = "Test City", state = "WA", zip = "12345", loyalty = true  • Sub-Case C: qty = 5, price = 88.50, name = "Test Customer", street = "123 Test St", city = "Test City", state = "OR", zip = "12345", loyalty = true  • Sub-Case D: qty = 9, price = 88.50, name = "Test Customer", street = "123 Test St", city = "Test City", state = "CA", zip = "12345", loyalty = true  • Sub-Case E: qty = 10, price = 70.00, name = "Test Customer", street = "123 Test St", city = "Test City", state = "WA", zip = "12345", loyalty = true  • Sub-Case F: qty = 11, price = 70.00, name = "Test Customer", street = "123 Test St", city = "Test City", state = "WA", zip = "12345", loyalty = true  • Sub-Case H: qty = 15, price = 70.00, name = "Test Customer", street = "123 Test St", city = "Test City", state = "WA", zip = "12345", loyalty = true  • Sub-Case I: qty = 19, price = 70.00, name = "Test Customer", street = "123 Test St", city = "Test City", state = "WA", zip = "12345", loyalty = true  • Sub-Case J: qty = 20, price = 70.00, name = "Test Customer", street = "123 Test St", city = "Test City", state = "WA", zip = "12345", loyalty = true  • Sub-Case K: qty = 21, price = 70.00, name = "Test Customer", street = "123 Test St", city = "Test City", state = "WA", zip = "12345", loyalty = true  • Sub-Case L: qty = 30, price = 70.00, name = "Test Customer", street = "123 Test St", city = "Test City", state = "WA", zip = "12345", loyalty = true  • Sub-Case M: qty = 40, price = 70.00, name = "Test Customer", street = "123 Test St", city = "Test City", state = "WA", zip = "12345", loyalty = true |
| **Steps** | 1. For each DataRow sub-case, instantiate a new Customer using the provided data via customer.SetInfo(name, street, city, state, zip, loyaltyClub).  2. Create an Order by calling the helper method TestOrder(qty, price, customer) with the specified quantity and price. 3. Create an Invoice using the order and customer. 4. Redirect the console output to a StringWriter to capture the printed invoice. 5. Call invoice.PrintInvoice() to output the invoice. 6. Capture the output string. 7. Verify that the output contains all required elements:  • "ACME Anvils Corporation"  • "Customer Invoice"  • "SHIP TO:"  • Customer address details (name, street, city, state, zip) exactly as provided  • "Quantity ordered:"  • "Cost per anvil:"  • "Subtotal:"  • "Less 15% Loyalty Club:" (this extra loyalty discount line must appear)  • "Taxable amount:"  • "Sales Tax:"  • "Shipping:"  • "TOTAL:" |
| **Expected Results** | For every sub-case, the printed invoice must include all of the above strings. In addition, the invoice for a loyalty member should show the loyalty discount line ("Less 15% Loyalty Club:") and the recalculated taxable amount ("Taxable amount:"). |
| **Actual Results** | All required substrings are contained in all datarow outputs |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Medium |
| **Comments** | This test validates that when a customer is a loyalty club member, the invoice includes additional discount information. It covers multiple scenarios (different order quantities, pricing tiers, and shipping states) to ensure that the invoice always shows the required header, address, order details, discount line, tax, shipping, and total. |

Clerk Tests

|  |  |
| --- | --- |
| **Test Case: Display\_ClerkName\_Whencalled** | |
| **ID** | **TST-CLRK-001** |
| **Unit Tested** | Display\_ClerkName\_Whencalled() |
| **Preconditions** | 1. MSTest environment is configured. 2. The Clerk class is compiled without errors. |
| **Environment** | .NET environment, MSTest framework, local development machine |
| **Test Data** | - Expected Clerk Name: "Bob" |
| **Steps** | 1. Instantiate a new Clerk object. 2. Call clerk.SetName("Bob") to set the clerk’s name. 3. Call clerk.GetName() to retrieve the name. 4. Compare the retrieved name with the expected value ("Bob"). |
| **Expected Results** | The call to GetName() returns the string "Bob", matching the expected value exactly. |
| **Actual Results** | The actual and expected are equal |
| **Pass/Fail** | Pass |
| **Priority** | High |
| **Severity** | Low |
| **Comments** | This test confirms that the Clerk class correctly stores and returns the name that is set via SetName(). |

Documentation of Executable Tests



Revision History

\*\*Please note that all text in the color of the current text you are reading is areas of the project I have revised as part of our 1.1 project.

I went about adding our new feature branch by first running the program and comparing it to our initial requirements given to us for the Acme Anvil Order Taking interview. I then read through the documentation provided and compared it to the interview notes and the program to verify it fulfilled the requirements. I noticed that there were some divergences such as formatting in the program and missed documentation sections. I worked to remedy this first so I could have a solid place to then add our new feature branches.

To start adding the feature branches I added the new features to our functional requirement documentation and then when through the document adding the additional detail necessary for the added functionality. Once that was complete, I moved onto the technical requirements. I added the new features to such as the updated pricing to the appropriate areas and added additional requirements for things such as the Futility Club membership discount. One the static tests and test plans were in documented, I started coding in the new features. Once I had finished coding the new features and manually tested them I started running the tests to see which has broken, requiring updates. I then started adding tests for any uncovered feature, adding in to existing tests new feature data as required, as well as adding new tests to directly tests the new features added. As I coded tests to verify code coverage was complete, I checked that each method on the order taking program was checked as covered by MSTest in visual studio.

Once all functionality was covered and the tests ran correctly a minimum of 15 times in repetition, I verified that all changes matched the documentation and completed any remaining test case documentation.

Review

1.) Your assessment of what was provided.

The SRS I received did have a few parts missing but the main issue I ran into was that the user stories were not done with single responsibilities, there were no acceptance criteria and many of the use cases were missing. I also did not originally have the executable tests.

2.) How did you deal with any ambiguities?

I spoke with Oscar to check in if there were executable tests which he did get me later that night. He was super helpful with getting me the tests quickly. Largely the ambiguities from the SRS were dealt with by comparing them to the Client interview documentation we all received. I then went through the document to make sure all original requirements were fulfilled.

3.) Did you need to add anything beyond the new features?

Due to the fact that many of the use cases were missing and I found that I needed to add quite a few parts. Everything I touched is in blue. If it was an edit, only the portion I edited is in blue, if I had to edit everything the whole section is blue along with the title. Many of the images needed to be redone due to the feature so they are all in blue.

Sections I edited:

|  |  |
| --- | --- |
| Edited | Added by Me |
| Scope  Use Case 1, 2, 3, 4  State Transition Diagram  Dependency Graph | Table of contents  Traceability  List of steps  User stories 4 & 5  Acceptance Criteria  Use Case 6, 7, 8  Test Cases  Executable Test documentation |

## ACME ANVILS ORDER TAKING

Version 1.0

Written by Oscar Moreno

## Business Statement / Functional Narrative / Scope

**What does the system do:**  
The ACME Anvils Order Taking app lets a clerk take a single order for anvils from a customer. It starts by asking the clerk’s name, shows a welcome message, collects customer shipping info and anvil quantity, calculates the cost (with tiered pricing, tax, and shipping), and prints a neatly formatted invoice. It’s built for one order per run—clerks restart it for each new order. This tool keeps ACME’s cartoon villain customers equipped with anvils, maintaining their 50+ year legacy.

Scope

* Collect the clerk’s name and customer order details.
* Calculate costs (subtotal, tax, shipping) and display an invoice matching the sample output.
* Handle one order per execution with pauses before invoice and exit.

## Assumptions

* The clerk enters valid data (no validation required for Version 1).
* Console environment supports basic formatting (spaces, newlines).
* Pricing: $88.50 (1–9), $70.00 (10–19), $68.25 (20+); Tax: 9.5%; Shipping: $112/anvil unless free (CA/OR, <5 anvils).
* Sample output formatting supersedes minor spec inconsistencies (e.g., prompt wording).

## User Stories

* **As a clerk**, I want to enter my name, so it appears on the invoice.
* **As a clerk**, I want a welcome message with stars and the motto for branding.
* **As a clerk**, I want to input order details with specific prompts on the same line as responses.
* **As a clerk**, I want an invoice showing all costs and shipping details in a boxed format.
* **As a clerk**, I want pauses to control when the invoice appears, and the program ends.

## Use Case Scenarios

**Use Case 1: Placing a Standard Order**

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched, and the clerk is ready to take an order.

**Main Flow:**

1. The system prompts the clerk for their name.
2. The system displays a welcome message with the ACME slogan.
3. The system asks for the number of anvils to order.
4. The clerk enters a valid number of anvils.
5. The system requests the customer's shipping details (name, address, city, state, zip).
6. The clerk enters the customer’s information.
7. The system calculates the total cost, including tax and shipping.
8. The system prompts the clerk to press a key to display the invoice.
9. The system displays the invoice with all order details.
10. The system thanks the user with a promotional slogan.
11. The system prompts the clerk to press a key to end the program.

**Postconditions:**

* The order details are displayed successfully.
* The clerk is prompted to restart a new order.

**Use Case 2: Ordering Less Than 5 Anvils in CA or OR (Free Shipping)**

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched, and the clerk is ready to take an order.
* The customer is located in **California (CA) or Oregon (OR)**.
* The customer orders fewer than **5 anvils**.

**Main Flow:**

1. The clerk enters a valid number of anvils (1-4).
2. The clerk provides the customer’s shipping address, including **CA or OR** as the state.
3. The system calculates the total cost, including tax but **waives shipping fees**.
4. The system displays a message: **"Congratulations! You qualify for FREE SHIPPING!"**.
5. The invoice is displayed with a shipping cost of **$0.00**.
6. The system thanks the user and ends the session.

**Postconditions:**

* The invoice reflects **free shipping**.
* The order total is correctly calculated.

**Use Case 3: Ordering 10+ Anvils for Discounted Pricing**

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched, and the clerk is ready to take an order.
* The customer orders **10 or more anvils**.

**Main Flow:**

1. The clerk enters a number **between 10 and 19**.
2. The system applies to the **discounted price of $70.00 per anvil**.
3. The system calculates subtotal, tax, and shipping fees.
4. The system generates the invoice with the discounted price per anvil.
5. The system displays the invoice and thanks the user.

**Postconditions:**

* The invoice reflects the correct discounted price.

**Use Case 4: Ordering 20+ Anvils for Bulk Discount**

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched, and the clerk is ready to take an order.
* The customer orders **20 or more anvils**.

**Main Flow:**

1. The clerk enters **20 or more anvils**.
2. The system applies to the **bulk discount of $68.25 per anvil**.
3. The system calculates subtotal, tax, and shipping fees.
4. The system generates the invoice with the bulk discount pricing.
5. The system displays the invoice and thanks the user.

**Postconditions:**

* The invoice reflects the correct bulk discount.

**Use Case 5: Invalid Data Entry (Optional - If Validation Is Implemented)**

**Actors:**

* Clerk (Primary Actor)

**Preconditions:**

* The system is launched.

**Main Flow:**

1. The clerk enters **an invalid number of anvils** (e.g., negative number, letters).
2. The system displays an **error message** and asks for a valid input.
3. The clerk enters **an invalid zip code** (e.g., non-numeric characters).
4. The system displays an **error message** and requests re-entry.
5. The system only proceeds once all inputs are valid.

**Postconditions:**

* The system handles errors properly and prevents invalid data submission.

## Test Plans & Supporting Bases

**1. Truth Tables (Logical Rules for Order Processing)**

A **Truth Table** represents different scenarios of free shipping eligibility.

|  |  |  |
| --- | --- | --- |
| **Anvils Ordered** | **State (CA/OR)** | **Free Shipping Applied?** |
| 1-4 | CA | Yes |
| 1-4 | OR | Yes |
| 1-4 | AZ | No |
| 5+ | CA | No |
| 5+ | OR | No |
| 10+ | Any State | No |

**2. Decision Table (Pricing and Tax Calculation Rules)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Condition** | **1-9 Anvils** | **10-19 Anvils** | **20+ Anvils** | **CA/OR & <5 Anvils** | **Shipping Charge** | **Tax (9.5%) Applied?** |
| Scenario 1: Order 3 anvils in CA | $88.50 each | - | - | Yes | $0.00 | Yes |
| Scenario 2: Order 10 anvils in AZ | - | $70.00 each | - | No | $112 per anvil | Yes |
| Scenario 3: Order 25 anvils in TX | - | - | $68.25 each | No | $112 per anvil | Yes |

**3. Boundary Value Analysis (BVA) Table**

**Testing boundary conditions for order quantity**:

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Input Value** | **Expected Outcome** |
| Min Order | 1 | Correct price tier applied |
| Just Below Discount | 9 | $88.50 per anvil |
| First Discount Threshold | 10 | $70.00 per anvil |
| Just Below Bulk Discount | 19 | $70.00 per anvil |
| Bulk Discount Threshold | 20 | $68.25 per anvil |
| Extreme Large Order | 1000 | System handles large numbers correctly |

**4. Equivalence Partitioning (EP) Table**

Dividing the order values into **equivalence classes** to test representative cases:

|  |  |  |  |
| --- | --- | --- | --- |
| **Partition** | **Valid/Invalid** | **Test Input** | **Expected Result** |
| Small Order | Valid | 3 anvils | Price = $88.50 per anvil |
| Mid-Tier Order | Valid | 15 anvils | Price = $70.00 per anvil |
| Bulk Order | Valid | 25 anvils | Price = $68.25 per anvil |
| Invalid Negative Order | Invalid | -1 anvils | Error Message |
| Non-Numeric Input | Invalid | "ten" anvils | Error Message |

**5. State Transition Diagram**

**States:**

1. **Start** → Clerk enters name
2. **Order Entry** → Clerk inputs number of anvils
3. **Customer Info** → Clerk enters shipping details
4. **Calculation** → System computes price, tax, and shipping
5. **Invoice Display** → Invoice is shown to the user
6. **End**

**Transitions:**

* **Valid input → next state**
* **Invalid input → retry state**
* **Final invoice display → end state**

**6. Dependency Graphs**

Illustrating dependency between different modules:

1. **Order Input** ⟶ **Price Calculation** ⟶ **Tax Calculation** ⟶ **Shipping Calculation** ⟶ **Invoice Generation**
2. Free Shipping Rule **depends** on **Order Input** and **State Selection**.
3. **Invoice Formatting** depends on all previous calculations.

(*Graph will be created to show these relationships.*)

# Technical Requirements

## Non-Functional Requirements

**1. Performance**

* **Response Time:** User input and calculations should be completed within **3 seconds**.
* **Throughput:** Support **100 orders per minute** without lag.

**2. Usability**

* **Ease of Use:** Guided order flow; completion within **2 minutes**.
* **Formatting:** Console output should be **readable, aligned, and intuitive**.
* **Learnability:** New users should operate efficiently within **15 minutes** of training.

**3. Availability & Reliability**

* **Uptime:** **99% availability** during business hours.
* **Error Handling:** Clear prompts for invalid inputs, ensuring **data integrity and fault tolerance**.

**4. Scalability**

* Support **10,000 orders per day** without performance issues.
* **Future-ready** for database integration and multi-order processing.

**5. Security**

* **Data Privacy:** No persistent storage of personal data without encryption.
* **Input Validation:** Prevent unexpected inputs and **injection attacks**.

**6. Maintainability**

* **Code Structure:** Modular, well-commented, and maintainable.
* **Error Logging:** Generate logs for troubleshooting.

**7. Portability**

* Compatible with **Windows, macOS, and Linux** with minimal dependencies.

**8. Compliance**

* **Tax Compliance:** Enforce **9.5% tax** calculation.
* **Financial Accuracy:** Dollar amounts formatted correctly with **two decimal places**.

**9. Testing & Completion Criteria**

* **Unit Test Coverage:** **90%+** for calculations and input handling.
* **System Testing:** All functional and non-functional tests must pass.
* **Load Testing:** Simulate **500 concurrent orders** for performance validation

## Entity Relationship Diagram

A screenshot of a computer

AI-generated content may be incorrect.

## Class Diagrams

A diagram of a computer

AI-generated content may be incorrect.

## Sequence Diagram

A diagram of a diagram

AI-generated content may be incorrect.

## Class Responsibility Collaboration

|  |  |
| --- | --- |
| **Clerk** | |
| Takes order from customer.  Inputs customer details.  Displays invoice to the customer. | OrderSystem (to start order processing)  Order (to store order details)  Customer (to store customer details) |

|  |  |
| --- | --- |
| **Customer** | |
| Stores shipping details.  Provides address validation (optional for later versions). | Order (order is associated with a customer) |

|  |  |
| --- | --- |
| **Order** | |
| Stores order details (number of anvils, subtotal, tax, total).  Calculates subtotal, tax, and total.  Determines shipping cost. | Pricing (for per-anvil pricing).  Customer (for shipping address).  Invoice (to generate the invoice). |

|  |  |
| --- | --- |
| **Pricing** | |
| Provides per-anvil price based on quantity.  Determines shipping cost rules (e.g., free shipping in CA/OR). | Order (to calculate costs). |

|  |  |
| --- | --- |
| **Invoice** | |
| Generates a formatted invoice with order details.  Displays invoice to the user. | Order (fetches order details). |

|  |  |
| --- | --- |
| **OrderSystem** | |
| Manages the overall flow of order processing.  Calls appropriate methods for calculations and invoice generation. | Clerk (user interaction).  Order (handles order details).  Invoice (for displaying final order summary). |

## Task Object Responsibility

|  |  |  |
| --- | --- | --- |
| **Task** | **Object** | **Responsibility** |
| Start order-taking process | OrderSystem | Initiate the order-taking session. |
| Capture clerk’s name | Clerk | Inputs the clerk’s name at the start. |
| Prompt for number of anvils | OrderSystem | Displays input prompt for quantity. |
| Capture customer details | Customer | Stores name and shipping address. |
| Determine price per anvil | Pricing | Provides pricing based on quantity tiers. |
| Calculate subtotal | Order | Multiplies quantity by price per anvil. |
| Apply sales tax | Order | tax calculations (9.5%) on subtotal. |
| Determine shipping cost | Order | Computers shipping fee based on location. |
| Check for free shipping | Pricing | Determines free shipping eligibility. |
| Generate invoice | Invoice | Formats and stores invoice details. |
| Display order summary | OrderSystem | Presents subtotal, tax, total, and shipping. |
| Display invoice to user | Invoice | Shows formatted invoice with all details. |
| Confirm order completion | Clerk | Finalizes the order after review. |
| End order session | OrderSystem | Ends execution after invoice confirmation. |

## State Diagrams

A diagram of a diagram

AI-generated content may be incorrect.