Store Loyalty System Documentation

1. System Overview

The Store Loyalty System is a comprehensive solution for managing customer purchases, loyalty points, and payment processing. The system handles both member and non-member customers, supporting different payment methods while maintaining a clear separation of concerns and following object-oriented design principles.

2. System Architecture

2.1 Core Components

- **Customer Management**: Handles member and non-member customers
- **Invoice Processing**: Manages purchase transactions
- **Payment Processing**: Supports cash and card payments
- **Loyalty Points**: Tracks points for member customers

2.2 Class Structure

- `Customer` (Abstract Base Class)
  - Defines common customer behavior
  - Manages invoice history
  - Abstract method for purchase processing
- `Member` (Concrete Class)
  - Implements loyalty points calculation
  - Tracks point accumulation
  - Processes member-specific purchase logic
- `NonMember` (Concrete Class)
  - Basic purchase processing
  - No loyalty points functionality
- `PaymentMethod` (Abstract Base Class)
  - Defines payment method interface
  - Ensures consistent payment processing
- `Invoice`
  - Stores transaction details
  - Links customer and payment information
- `StoreSystem`
  - Main system facade

- Coordinates all system components
- Manages customer registration and purchase processing

3. Implementation Details

3.1 Design Patterns Used

1. **Strategy Pattern**
   - Used for payment method implementation
   - Allows easy addition of new payment types
   - Encapsulates payment processing logic
2. **Abstract Factory**
   - Customer creation and management
   - Flexible customer type handling
3. **Facade Pattern**
   - StoreSystem provides simplified interface
   - Encapsulates complex subsystem interactions

3.2 Key Features

1. **Modular Design**
   - Each class has single responsibility
   - Easy to maintain and extend
   - Clear separation of concerns
2. **Robust Validation**
   - Input validation for all operations
   - Comprehensive error handling
   - Detailed logging for troubleshooting
3. **Flexible Architecture**
   - Easy to add new customer types
   - Extensible payment method system
   - Scalable points calculation

4. Testing Strategy

4.1 Test Coverage

- Unit tests for all major components
- Integration tests for system workflows
- Edge case handling
- Error condition testing

4.2 Test Cases

1. Customer Registration
2. Purchase Processing
3. Points Calculation
4. Payment Method Handling
5. Error Handling
6. Edge Cases

5. System Usage

5.1 Basic Operations

```
# Initialize system
store = StoreSystem()

# Register customer
customer = store.register_customer(1111, True)  # Member

# Process purchase
result = store.process_purchase(1111, 1, 810, "Card")
```

5.2 Common Workflows

1. Member Purchase
2. Non-member Purchase
3. Points Calculation
4. Payment Processing

6. Maintenance and Extension

6.1 Adding New Features

- Create new customer types
- Add payment methods
- Modify points calculation
- Extend validation rules

6.2 System Monitoring

- Logging system in place
- Transaction tracking
- Error monitoring
- Performance metrics

7. Conclusions and Recommendations

7.1 System Strengths

- Robust architecture
- Comprehensive testing
- Clear documentation
- Extensible design

7.2 Future Improvements

- Database integration
- API development
- User interface
- Advanced reporting