Durable_rules 재고관리시스템

2024254012 배인호

물류 창고 실제 재고를 기반

재고의 입출고 및 이동 등에 대한 규칙을 정함

이전 페이지 | **7. 플레이우드메이커** | 다음 페이지

날짜	월	11			12			1			2			3			4			합계		
제품명	재고	입고	출고	잔고	입고	출고	잔고	입고	출고	잔고	입고	출고	잔고	입고	출고	잔고	입고	출고	잔고	입고	출고	재고
3월-TV무드등	71			71		30	41		41	0			0			0			0	0	71	**0**
오리스쿠터	506			506		30	476		41	435			435			435			435	0	71	**435**
프로펠러 비행기	657			657		375	282		41	241			241			241			241	0	416	**241**
엘리베이터	727			727		403	324		61	263			263			263			263	0	464	**263**
양날개 프로펠러 비행기	655			655			655			655			655			655			655	0	0	**655**
회전목마	800		86	714		98	616		13	603		88	515			515			515	0	285	**515**
대포	800			800		48	752		13	739		88	651			651			651	0	149	**651**
천칭레이더	800			800		48	752		13	739		88	651			651			651	0	149	**651**
페널티킥	800			800		48	752		13	739		88	651			651			651	0	149	**651**
다람쥐	800			800			800			800		3	797			797			797	0	3	**797**
합계	6,616	0	86	6,530	0	1,080	5,450	0	236	5,214	0	355	4,859	0	0	4,859	0	0	4,859	0	1,757	**4,859**

날짜	품명	주소	출고량
2024-02-26			

- 규칙

1. 재고부족 2. 과다재고 3. 재고 판매 4. 재고 반품 5. 알림 6. 물품이동
7. 판매기간 경과 8. MOQ할인 9. 주문처리 10. 반품처리 11. 이동처리
12. 할인처리 13. 신제품 홍보 14. 입고 알림 15. 인기제품관리
16.선 구매 특전 17. 이벤트 18. 손실관리 19. 시즌 관리

재고 데이터를 넣어 규칙 기반에 사실 주장을 해봄

```
'20색 필라멘트': 360,
'크리스탈 대': 752,
'크리스탈 대-패드': 1102,
'크리스탈 대-스펀지': 1000,
'크리스탈 소': 1185,
'크리스탈 소-패드': 1018,
'크리스탈 소-스펀지': 948,
'키모도장 블루': 470,
'키모도장 핑크': 101,
'키모도장 퍼플': 0,
'키모도장-패드': 661,
'키모도장-스펀지': 1073,
```

```python
# 재고 부족 알림
@when_all((m.quantity < 300) & (m.quantity > 0))
def low_stock(c):
    c.assert_fact({'type': 'alert', 'message': '재고 부족 - 품목 {}의 재고가 부족합니다'.format(c.m.item_id)})
    print('재고 부족 알림 - 품목 {}의 재고 부족'.format(c.m.item_id))

# 과다 재고 알림
@when_all(m.quantity >= 1000)
def excess_stock(c):
    c.assert_fact({'type': 'alert', 'message': '과다 재고 - 품목 {}의 재고가 과다합니다'.format(c.m.item_id)})
    print('과다 재고 알림 - 품목 {}의 재고 과다'.format(c.m.item_id))

# 재고가 판매될 때
@when_all((m.action == 'sell') & (m.quantity > 0))
def process_order(c):
    c.assert_fact('inventory_management', {'item_id': c.m.item_id, 'quantity': -1 * c.m.quantity, 'action': 'update'})
    print('{} 개의 품목 {}이 판매되었습니다'.format(c.m.quantity, c.m.item_id))

# 재고가 반품될 때
@when_all((m.action == 'return') & (m.quantity > 0))
def return_stock(c):
    c.assert_fact('inventory_management', {'item_id': c.m.item_id, 'quantity': c.m.quantity, 'action': 'update'})
    print('{} 개의 품목 {}이 반품되었습니다'.format(c.m.quantity, c.m.item_id))

# 재고 부족시 알림
@when_all((m.quantity == 0))
def out_of_stock(c):
    c.assert_fact({'type': 'alert', 'message': '재고 부족 - 품목 {}의 재고가 없습니다'.format(c.m.item_id)})
    print('재고 부족 알림 - 품목 {}의 재고 부족'.format(c.m.item_id))

# 재고 충분시 알림
@when_all((m.quantity >= 500) & (m.quantity < 1000))
def sufficient_stock(c):
    c.assert_fact({'type': 'alert', 'message': '품목 {}의 재고가 충분합니다'.format(c.m.item_id)})
    print('충분한 재고 알림 - 품목 {}의 재고 충분'.format(c.m.item_id))

# 재고 이동시
@when_all((m.action == 'move') & (m.quantity > 0))
def move_stock(c):
    print('{} 개의 품목 {}이 이동되었습니다'.format(c.m.quantity, c.m.item_id))

# 판매 기간 경과 알림
@when_all((m.expiry_date < '2024-12-31') & (m.quantity > 0))
def expiry_alert(c):
    print('유통 기한 경과 - 품목 {}의 판매 기한이 지났습니다'.format(c.m.item_id))

# MOQ할인 - 100개 이상 구매시
@when_all((m.action == 'discount') & (m.quantity >= 100))
def moq_discount(c):
    print('MOQ 할인 적용 - 품목 {}의 주문이 100개 이상으로 할인율이 적용됩니다'.format(c.m.item_id))
```

좌측 결과 화면

```
50 개의 품목 크리스탈 대-패드이 반품되었습니다
20 개의 품목 크리스탈 대-패드이 이동되었습니다
20 개의 품목 크리스탈 대-패드이 이동됨
재고 이동 로그 기록 - 품목 크리스탈 대-패드의 재고가 이동됨
유통 기한 경과 - 품목 3월-TV무드등의 판매 기한이 지났습니다
MOQ 할인 적용 - 품목 Stampmaker장비(P90)의 주문이 100개 이상으로 할인율이 적
할인 처리 - 특정 제품 세트를 구매하면 할인을 제공합니다
20 개의 품목 퍼즐 Pre-A4이 반품되었습니다
100 개의 품목 청사초롱이 이동되었습니다
100 개의 품목 청사초롱이 이동됨
재고 이동 로그 기록 - 품목 청사초롱의 재고가 이동됨
MOQ 할인 적용 - 품목 베이직도장-패드의 주문이 100개 이상으로 할인율이 적용됩니
할인 처리 - 특정 제품 세트를 구매하면 할인을 제공합니다
최신 제품 홍보 시작 - 새로운 제품이 출시되면 관련된 마케팅 활동을 시작합니다
30 개의 품목 베이직도장-뚜껑이 이동되었습니다
30 개의 품목 베이직도장-뚜껑이 이동됨
재고 이동 로그 기록 - 품목 베이직도장-뚜껑의 재고가 이동됨
입고 예정 알림 - 입고 예정인 제품에 대한 알림을 관리자에게 전송합니다
인기 제품 재고 관리 - 인기 있는 제품에 대한 재고를 신속하게 관리하여 수요를 충
선구매 특전 제공 - 새로운 제품 출시 전 선구매자에게 특전을 제공합니다
제품 할인 이벤트 - 특정 기간 동안 일부 제품에 할인 이벤트를 진행합니다
재고 손실 관리 - 재고 손실 발생 시 해당 사항을 추적하고 조치를 취합니다
시즌 제품 재고 관리 - 시즌에 따라 수요가 변하는 제품의 재고를 관리합니다
```

사실 주장

```
# 재고가 판매될 때 테스트 (크리스탈 대)
assert_fact('inventory_management', {'item_id': '크리스탈 대', 'quantity': 100, 'action': 'sell'})

# 재고가 반품될 때 테스트 (크리스탈 대-패드)
assert_fact('inventory_management', {'item_id': '크리스탈 대-패드', 'quantity': 50, 'action': 'return'})

# 재고 부족시 알림 테스트 (키잼펜)
assert_fact('inventory_management', {'item_id': '키잼펜', 'quantity': 0})

# 재고 충분시 알림 테스트 (20색 필라멘트)
assert_fact('inventory_management', {'item_id': '20색 필라멘트', 'quantity': 800})

# 재고 이동시 테스트 (크리스탈 대-패드)
assert_fact('inventory_management', {'item_id': '크리스탈 대-패드', 'quantity': 20, 'action': 'move'})

# 판매 기간 경과 알림 테스트 (3월-TV무드등)
assert_fact('inventory_management', {'item_id': '3월-TV무드등', 'quantity': 50, 'expiry_date': '2024-03-15

# MOQ할인 테스트 (Stampmaker장비(P90))
assert_fact('inventory_management', {'item_id': 'Stampmaker장비(P90)', 'quantity': 120, 'action': 'discou

# 주문 처리 테스트 (퍼즐 pre-A1)
assert_fact('inventory_management', {'item_id': '퍼즐 pre-A1', 'quantity': 50, 'action': 'sell'})

# 반품 처리 테스트 (퍼즐 Pre-A4)
assert_fact('inventory_management', {'item_id': '퍼즐 Pre-A4', 'quantity': 20, 'action': 'return'})

# 이동 처리 테스트 (청사초롱)
assert_fact('inventory_management', {'item_id': '청사초롱', 'quantity': 100, 'action': 'move'})

# 할인 처리 테스트 (베이직도장-패드)
assert_fact('inventory_management', {'item_id': '베이직도장-패드', 'quantity': 150, 'action': 'discount'})

# 최신 제품 홍보 테스트 (Stampmaker장비(P90))
assert_fact('inventory_management', {'item_id': 'Stampmaker장비(P90)', 'action': 'promotion'})
```

코드 수정

```
assert_fact('business_rules_new', {'action': 'inventory', 'item_count': 250, 'item_name': 'A'})
assert_fact('business_rules_new', {'action': 'inventory', 'item_count': 1200, 'item_name': 'B'})
assert_fact('business_rules_new', {'action': 'inventory', 'item_count': 0, 'item_name': 'C'})
assert_fact('business_rules_new', {'action': 'sales', 'item_name': 'D', 'item_count': 50})
assert_fact('business_rules_new', {'action': 'return', 'item_name': 'E', 'item_count': 10})
assert_fact('business_rules_new', {'action': 'movement', 'item_name': 'F', 'item_count': 30})
assert_fact('business_rules_new', {'action': 'expired', 'item_name': 'G'})
assert_fact('business_rules_new', {'action': 'purchase', 'item_name': 'H', 'item_count': 150})
assert_fact('business_rules_new', {'action': 'new_item', 'item_name': 'I', 'item_count': 200})
assert_fact('business_rules_new', {'action': 'purchase', 'item_name': 'J', 'item_count': 350})
assert_fact('business_rules_new', {'action': 'purchase', 'item_name': 'K', 'item_count': 100, 'customer_type': 'new'})
assert_fact('business_rules_new', {'action': 'purchase', 'item_name': 'L', 'item_count': 200, 'customer_type': 'vip'})
assert_fact('business_rules_new', {'action': 'seasonal_demand', 'item_name': 'M'})
assert_fact('business_rules_new', {'action': 'pre_order', 'item_name': 'N'})
```

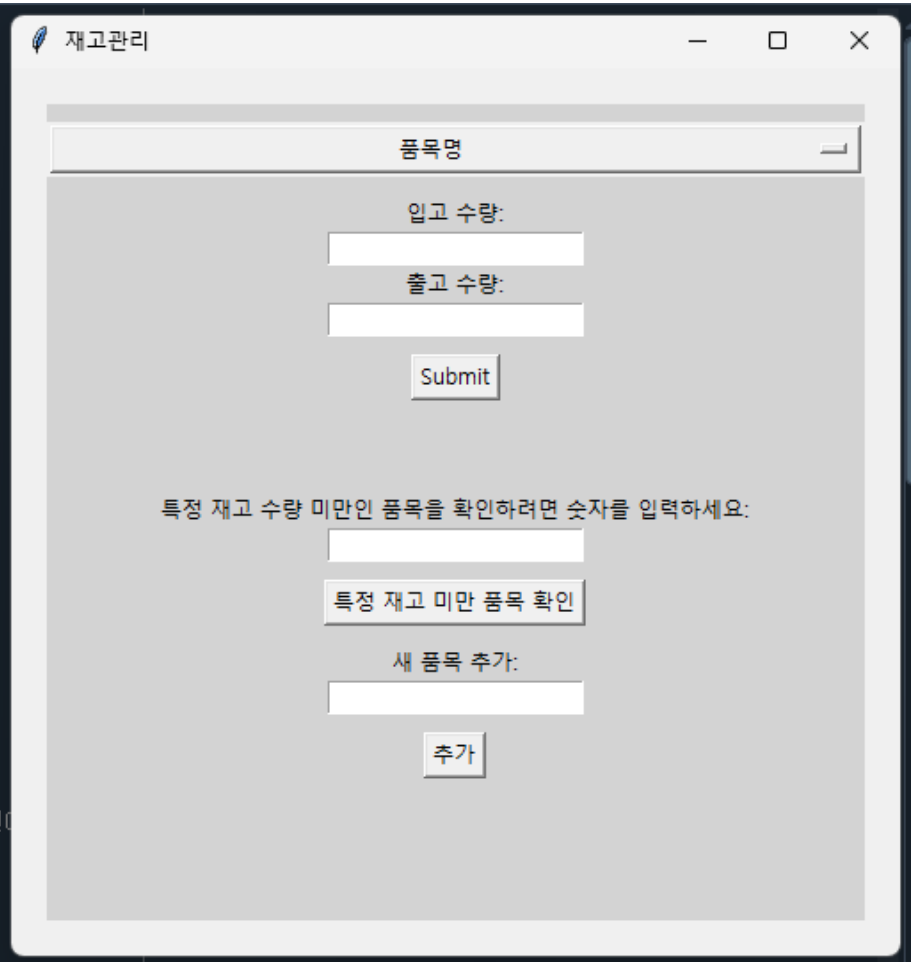
```
규칙1: 품목 A의 재고가 부족합니다.
규칙2: 품목 B의 재고가 과다합니다.
규칙1: 품목 C의 재고가 부족합니다.
규칙3: 품목 C의 재고가 없습니다.
규칙4: 품목 D이(가) 50개가 판매되었습니다.
규칙5: 품목 E이(가) 10개가 반품되었습니다.
규칙6: 품목 F이(가) 30개가 이동되었습니다.
규칙7: 품목 G의 판매기간이 지났습니다.
규칙8: 품목 H을(를) 한번에 100개 이상 구매하면 10% 할인됩니다.
규칙10: 새로운 품목 I이(가) 200개 입고되어 프로모션을 시작합니다.
규칙8: 품목 J을(를) 한번에 100개 이상 구매하면 10% 할인됩니다.
규칙9: 품목 J을(를) 한번에 200개 이상 구매하면 20% 할인됩니다.
규칙11: 품목 J을(를) 한번에 300개 이상 구매하면 30% 할인됩니다.
규칙8: 품목 K을(를) 한번에 100개 이상 구매하면 10% 할인됩니다.
규칙12: 신규고객이 구매시 무료배송 제공됩니다.
규칙8: 품목 L을(를) 한번에 100개 이상 구매하면 10% 할인됩니다.
규칙9: 품목 L을(를) 한번에 200개 이상 구매하면 20% 할인됩니다.
규칙13: VIP고객이라면 1+1 혜택이 제공됩니다.
규칙14: 시즌에 따라 수요가 변하는 품목 M은(는) 이동처리할 수 있도록 해주십시오.
규칙15: 선주문을 넣은 사람에게는 특전을 제공합니다.
```

```
from durable.lang import *

# 새로운 규칙 세트 생성
with ruleset('business_rules_new'):

    # 규칙 1: 품목이 300개 이하라면 품목의 재고가 부족하다 알려줌
    @when_all((m.action == 'inventory') & (m.item_count <= 300))
    def low_inventory(c):
        print('규칙1: 품목 {}의 재고가 부족합니다.'.format(c.m.item_name))

    # 규칙 2: 품목이 1000개 이상이라면 품목의 재고가 과다하다 알려줌
    @when_all((m.action == 'inventory') & (m.item_count >= 1000))
    def excessive_inventory(c):
        print('규칙2: 품목 {}의 재고가 과다합니다.'.format(c.m.item_name))

    # 규칙 3: 품목 0이 된다면 품목의 재고가 없다고 알려줌
    @when_all((m.action == 'inventory') & (m.item_count == 0))
    def zero_inventory(c):
        print('규칙3: 품목 {}의 재고가 없습니다.'.format(c.m.item_name))

    # 규칙 4: 품목이 판매가 된다면 개수를 알려줌
    @when_all((m.action == 'sales'))
    def sales_notification(c):
        print('규칙4: 품목 {}이(가) {}개가 판매되었습니다.'.format(c.m.item_name, c.m.item_count))

    # 규칙 5: 품목이 반품이 되었다면 개수를 알려줌
    @when_all((m.action == 'return'))
    def return_notification(c):
        print('규칙5: 품목 {}이(가) {}개가 반품되었습니다.'.format(c.m.item_name, c.m.item_count))

    # 규칙 6: 품목이 이동한다면 개수를 알려줌
    @when_all((m.action == 'movement'))
    def movement_notification(c):
        print('규칙6: 품목 {}이(가) {}개가 이동되었습니다.'.format(c.m.item_name, c.m.item_count))

    # 규칙 7: 품목이 5개월의 기간이 지났다면 알려줌
    @when_all((m.action == 'expired'))
    def expired_notification(c):
        print('규칙7: 품목 {}의 판매기간이 지났습니다.'.format(c.m.item_name))

    # 규칙 8: 품목을 한번에 100개 이상 구매한다면 10% 할인율을 적용
    @when_all((m.action == 'purchase') & (m.item_count >= 100))
    def discount_10_percent(c):
        print('규칙8: 품목 {}을(를) 한번에 100개 이상 구매하면 10% 할인됩니다.'.format(c.m.item_name))

    # 규칙 9: 품목을 한번에 200개 이상을 구매한다면 20% 할인율을 적용
    @when_all((m.action == 'purchase') & (m.item_count >= 200))
    def discount_20_percent(c):
        print('규칙9: 품목 {}을(를) 한번에 200개 이상 구매하면 20% 할인됩니다.'.format(c.m.item_name))

    # 규칙 10: 새로운 품목이 입고된다면 프로모션 시작
    @when_all((m.action == 'new_item'))
    def promotion_start(c):
        print('규칙10: 새로운 품목 {}이(가) {}개 입고되어 프로모션을 시작합니다.'.format(c.m.item_name, c.m.item_count))

    # 규칙 11: 품목을 한번에 300개 이상을 구매한다면 30% 할인율을 적용
    @when_all((m.action == 'purchase') & (m.item_count >= 300))
    def discount_30_percent(c):
        print('규칙11: 품목 {}을(를) 한번에 300개 이상 구매하면 30% 할인됩니다.'.format(c.m.item_name))

    # 규칙 12: 신규고객이 구매시 무료배송 제공
    @when_all((m.action == 'purchase') & (m.customer_type == 'new'))
    def free_shipping(c):
        print('규칙12: 신규고객이 구매시 무료배송 제공됩니다.')

    # 규칙 13: VIP고객이라면 1+1 혜택 제공
    @when_all((m.action == 'purchase') & (m.customer_type == 'vip'))
    def buy_one_get_one_free(c):
        print('규칙13: VIP고객이라면 1+1 혜택이 제공됩니다.')

    # 규칙 14: 시즌에 따라 수요가 변하는 품목은 이동처리
    @when_all((m.action == 'seasonal_demand'))
    def handle_seasonal_demand(c):
        print('규칙14: 시즌에 따라 수요가 변하는 품목 {}은(는) 이동처리할 수 있도록 해주십시오.'.format(c.m.item_name))

    # 규칙 15: 선주문을 넣은 사람에게는 특전을 제공
    @when_all((m.action == 'pre_order'))
    def pre_order_benefit(c):
        print('규칙15: 선주문을 넣은 사람에게는 특전을 제공합니다.')
```

과제 중 실제 회사에 필요한 재고관리 시스템 필요성 인지

Thinter Messagebox를 활용하여 재고관리에 활용할 수 있는 툴을 만들어 봄

```python
import tkinter as tk
from tkinter import messagebox

# 초기 재고 딕셔너리
initial_stock = {item: 0 for item in ["품목명"]}

# 이전 재고 딕셔너리 초기화
previous_total_stock = initial_stock.copy()

# 파일에 재고 저장
def save_stock():
    with open("stock_data.txt", "w") as file:
        for item, stock in previous_total_stock.items():
            file.write(f"{item}:{stock}\n")

# 파일에서 재고 불러오기
def load_stock():
    try:
        with open("stock_data.txt", "r") as file:
            for line in file:
                item, stock = line.strip().split(":")
                previous_total_stock[item] = int(stock)
    except FileNotFoundError:
        pass

# 프로그램 실행 시 저장된 재고 데이터 불러오기
load_stock()

def on_submit():
    global previous_total_stock  # 함수 내에서 전역 변수를 사용하기 전

    selected_item = item_var.get()

    # 입고 및 출고 수량 입력 확인
    in_stock_entry = entry_in_stock.get()
```



윈도우에서 실제 실행 화면

재고 확인 및 입고 출고 반영 가능

Pyinstaller를 통하여 exe파일 형태로 실행