JAVA 음료 자동 판매기(Auto Vending Machin) 설계서

학과	컴퓨터소프트웨어공학과	이름	왕성훈	학번	20243522
담당 교수님	김석훈 교수님	과목	JAVA 프로그래밍	제출일	2025.06.20

상위 설계

상위 설계서를 통하여 전반적인 구조를 구성하고, 자판기의 효율적인 동작을 이루고자 한다. 개발하고자 하는 자판기는 파일 입출력을 통하여 하나의 기기에서 데이터를 저장하여 작동하는 방식을 사용할 수 있다.

1. 모듈

요구사항에 따르면, 자판기는 크게 음료 저장부, 화폐 저장부, 관리자 관리 부분으로 나누어질 수 있고, 음료 저장부는 8 종류의 음료를 저장하고, 기본적으로 각각의 음료 별 10 개의 음료를 저장한다. 화폐 저장부는 10, 50, 100, 500, 1000 원 단위로 화폐를 저장하고, 각 화폐별 10 개를 저장한다. 관리자 관리 부분에서는 수금 부분, 데이터 산출 부분, 관리자 접근 관리 부분 등이 있으며, 위 기능은 모두 사용자 인터페이스 환경에서 작동함으로, GUI 담당 부분이 필요하다. 위 사항들은 모듈로 구성이 가능하며, 비슷한 구성으로 모듈화 하여 설계하였다.

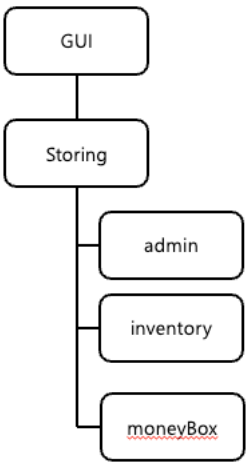


그림 1: 모듈 구성

GUI 는 사용자 인터페이스 기능을 담당하고, storing 모듈은 자판기의 전반적인 기능을 담당한다. Admin 은 관리자 페이지 접속, 관리자 정보 기록 및 저장, 데이터 산출 등의 역할을 하며, inventory 는 각 음료 종류별 음료를 저장한다. moneyBox 는 각 화폐별 화폐를 저장하는 부분이다.

2. 데이터

파일로 저장하는 데이터는 일별/월별 판매기록, 관리자 정보, 수입 정보 등 이고, 이 파일들은 각각이 저장하는 정보에 따라 다른 모듈에 의하여 관리되도록 구성하고자 한다.

다음으로, Inventory 데이터는 각 음료 별 음료들을 객체로 저장하는 방식을 채택하였다. 따라서, 음료는 하나의 클래스로 정보가 구성되며, 연결 리스트 구조를 통하여 음료가 인스턴스화 되어 저장되는 방식이다. 또한, 음료를 종류별로 연결 리스트로 저장한 뒤, 해당 종류별 연결 리스트를 다른 연결 리스트로 다시 묶어 2 차원 연결 리스트와 비슷한 개념으로 작동되게 할 전망이다. 이는 실제 자판기의 구성에서 착안한 설계로, 실제 자판기 또한 음료별 저장 부분이 있고, 그 저장 부분의 묶음이 하나의 보관함을 이루기 때문이다.

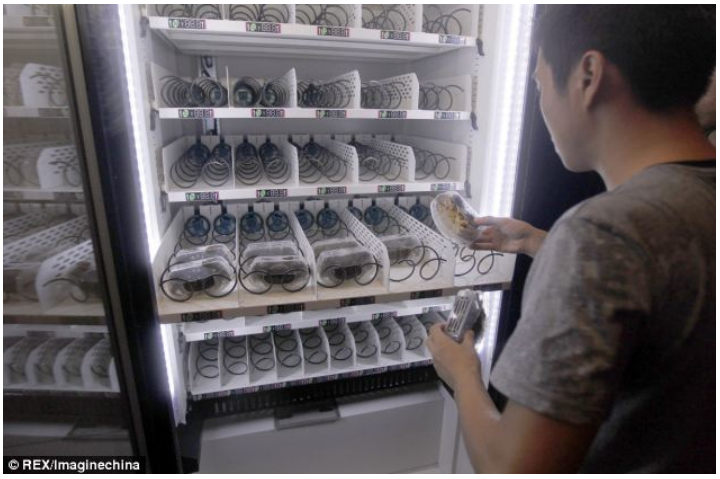


사진 2: 실제 자판기 내부 모습 예 [1]

위 자판기 사진처럼, 하나의 음료 종류에 대해 여러개의 음료가 저장되어 있을 때, 하나의 음료 종류를 보관하는 곳을 tray 라고 지칭하였다. 이 트레이는 요구사항에 의하여 8 개가 필요하고, 검색이 빈번할 것 같아 보여 inventory(전체보관함)에 이진 트리 형식으로 저장하는 방식을 고안하였다.

3. GUI 레이아웃

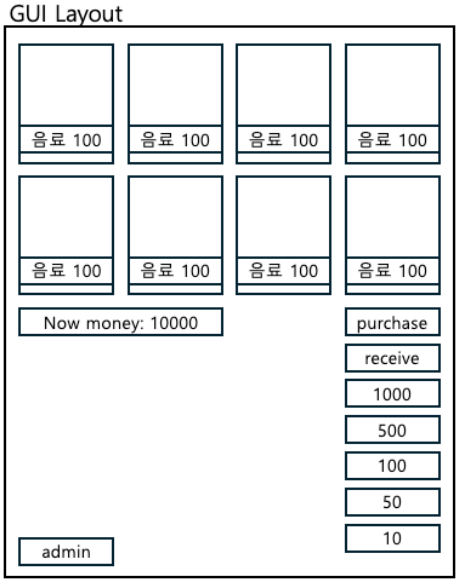


사진 3: GUI 레이아웃 구상도

위 사진과 같이 사용자 인터페이스를 설계하였다. 디자인은 실제 자판기를 참고하였고, 요구명세서에 따라 음료 8 개, 화폐는 100 원, 500 원, 100 원, 50 원, 10 원을 입력받을 수 있으며, 잔돈 회수 버튼인 receive, 구매 버튼인 purchase, 관리자 접근 버튼인 admin 으로 구성되어 있다.

하위 설계서

상위 설계서를 기반으로 하위 설계서를 구체적으로 작성함으로써, 추후 진행하게 될 자동판매기의 구성을 일관된 구조로 유지되도록 하고자 한다.

1. 설계 원칙 [2]

- SRP(Single Responsibility Principle): 단일 책임 원칙

단일 책임 원칙은, 코드의 유지보수성과 수정을 용이하게 하고, 응집도를 높이기 위하여 고안된 설계 원칙으로, 모든 클래스는 오직 하나의 책임만을 가져야 한다는 원칙이다. 여기서 책임이란, 클래스가 담당하는 기능을 의미한다.

이 원칙에 따르면, 클래스는 자신이 맡은 역할을 캡슐화해야 하며, 클래스가 제공하는 모든 기능은 오직 그 책임과만 관계가 있어야 한다. 예를 들어, 하나의 클래스가 동시에 다른 역할을 수행한다면, 이 클래스는 단일 책임 원칙이 위배된다.

단일 책임 원칙이 적용되지 않는다면, 하나의 기능을 수정할 때, 다른 여러가지의 코드를 함께 수정해야 한다는 비효율성이 발생하기 때문에 단일 책임 원칙은 중요하다고 판단할 수 있다.

- OCP(Open-Closed Principle): 개방-폐쇄 원칙

개방-폐쇄 원칙은 두 가지 의미를 갖는다.
1. 확장에 대해 열려 있다.(Open for extension):
새로운 요구사항이나 기능이 추가될 때, 기존의 코드를 변경하지 않아도 새로운 코드를 추가해 기능을 확장할 수 있다, 시스템의 동작을 쉽게 변경, 추가가 가능하다.
2. 변경에 대해 닫혀 있다.(Closed for modification):
기존의 코드를 수정하지 않아도 새로운 기능을 추가할 수 있다, 기존에 잘 동작하던 코드가 영향을 받지 않는다.

- LSP(Liskov Substitution Principle): 리스코프 치환 원칙 [3]

“자식 클래스는 언제나 부모 클래스로 교체할 수 있어야 한다.”는 원칙이다. 즉, 부모 클래스의 인스턴스를 사용하는 모든 곳에 자식 클래스를 대신 넣더라도, 프로그램의 동작이 변하지 않아야 한다.
예를 들어, 사각형과 정사각형의 관계에서, 정사각형이 사각형을 상속받는다고 가정하면, 메서드를 오버라이드할 때, 너비와 높이가 항상 같아야 하는 특성 때문에 부모가 기대하는 기능인 “너비와 높이를 독립적으로 변경 가능하다.” 가 깨질 수 있다.
이 원칙을 지키면, 코드의 유지보수성과 확장성이 높아질 수 있다. 위배한다면, 상속 구조에서 예기치 않은 버그나 오류가 발생할 수 있다.

2. 클래스 다이어그램을 통한 클래스, 자료구조 설계

클래스 다이어그램이란 클래스 내부의 구성요소를 표기하고, 클래스 간의 관계를 도식화 함으로써 프로그램의 구성을 표시할 수 있는 단순화 방법이다. 클래스 다이어그램을 표기하는

방법 중 UML은 표준화된 표기 방법이다. 위 설계 원칙에 최대한 맞추어 설계하려 노력해보았다.

- UML의 간단한 설명 [4] [5]

클래스는 보통 3개의 구획으로 나뉜 사각형으로 그리며, 가장 위에는 클래스 이름을, 그 아래에는 속성, 마지막에는 메서드를 표기한다. 클래스 이름은 반드시 표기해야 하며, 속성과 오퍼레이션은 필요에 따라 생략이 가능하다.

속성은 "접근제어자 필드명: 타입 = 기본값" 형식으로 표기하며, 오퍼레이션은 "접근제어자 메서드명(파라미터: 타입): 반환타입" 형식으로 작성한다. 접근제어자는 public은 +, private은 -, protected는 #, package는 ~로 나타낸다. 속성이나 오퍼레이션에 {readonly}를 붙이면 final, 밑줄을 긋거나 {static}을 붙이면 static임을 의미한다. 다중성은 *, 0..1 등으로 표기하여 리스트나 옵션 타입을 표현할 수 있다.

추상 클래스는 클래스 이름을 이탤릭체 또는 {abstract}를 클래스명 옆에 붙여서 나타낼 수 있다. 인터페이스는 클래스명 위에 <>와 같은 작은 글씨로 명시하여 표기한다.

클래스 간의 관계는 다양한 선과 기호로 표현한다. 연관관계(Association)는 실선으로 표기하고, 방향성이 있으면 화살표를 추가한다. 상속(Generalization)은 실선으로 표기하고, parent, child와 같은 텍스트로 표기할 수 있으며, 인터페이스 구현(Realization)은 점선 화살표로 표기한다. 집합관계(Aggregation)는 빈 다이아몬드, 합성관계(Composition)는 채워진 다이아몬드로 연결한다. 의존관계(Dependency)는 점선 화살표로 나타내며, 메서드의 파라미터나 반환 타입에 해당 클래스가 사용될 때 사용한다.

연관관계의 끝에는 다중성과 역할명을 표기할 수 있으며, 이는 클래스가 참조하는 인스턴스의 개수와 역할을 명확하게 하기 위함이다. 또한, 연관클래스(Association Class)를 사용하여 두 클래스 간의 관계에 추가적인 속성이 필요할 때 별도의 클래스로 정의할 수 있다.

클래스 다이어 그램 작성 방법을 참고하여, 부족하지만, 그림 1로부터 구성된 모듈의 구조에 맞게, 아래 사진들과 같이 클래스들을 각각 설계하였다.

- Inventory

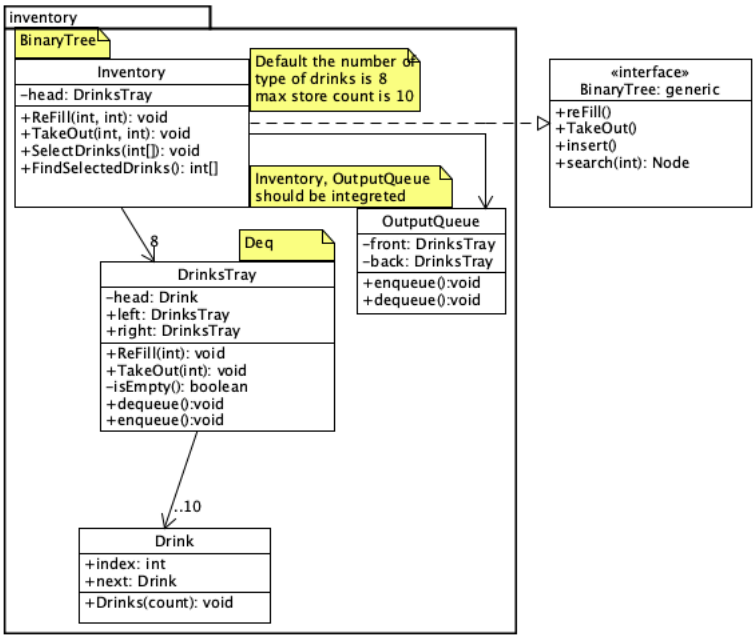


그림 3: Inventory 모듈의 클래스 다이어그램

모듈 inventory 같은 경우, 각 음료 종류 별 트레이를 이진 트리 구조로 저장한다. 이는 사용자와 상호작용 하면서 발생하는 빈번한 탐색에 대비하여 이러한 구조를 선택하게 되었다. 기본적으로 이진 트리 구조인 Inventory 클래스는, head 를 가지고 있고, DrinksTray 를 추가하는 refill(), 빼는 takeOut(), 선택하는 selectDrinks(), 탐색하는 findSelected() 메서드를 가지고 있다. Inventory 의 노드가 되는 DrinksTray 클래스는 연결리스트 기반 덱 구조이고, Drink 를 노드로 한다. 따라서, Drink 노드의 헤드, Inventory 가 이진 트리 이므로, left, right 를 가지며, 삽입, 삭제, enqueue, dequeue 기능이 있다.
OutputQueue 클래스 같은 경우는 사용자가 음료를 선택하면 큐에 저장하는 형식이고, 음료를 반출할 때에는 큐 방식으로 음료를 배출하게 되는데, 이 기능을 담당하게 된다.

- moneyBox

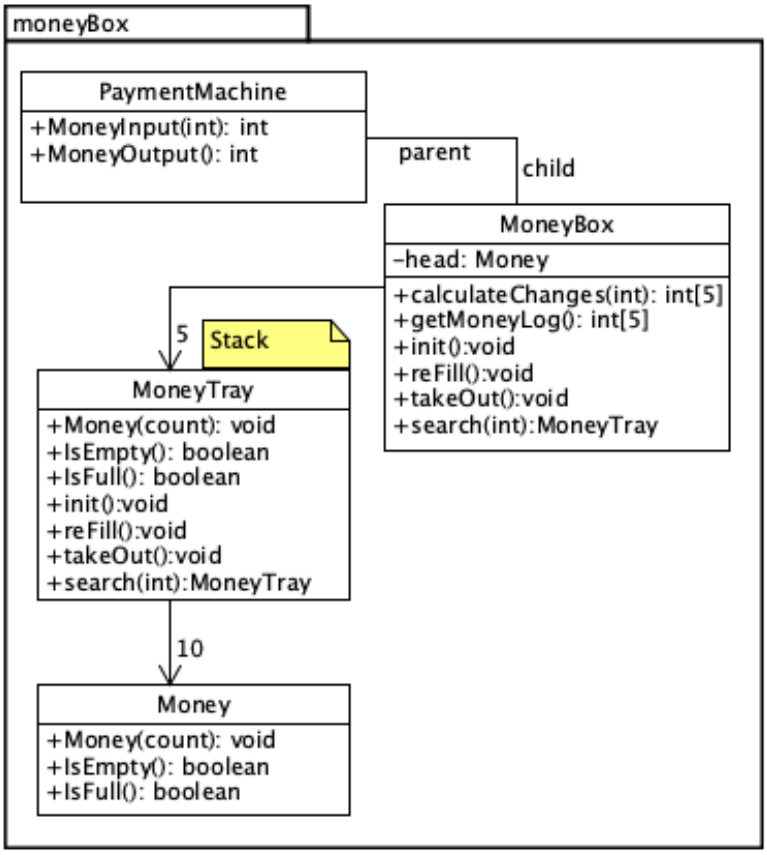


그림 4: moneybox 모듈의 클래스 다이어그램

moneyBox 같은 경우 연결리스트 구조를 통하여 각 화폐를 단위별로 저장하고 있는 moneyTray 를 저장하고, moneyTray 는 해당 단위의 화폐를 저장한다. 예를 들어, 100 원을 저장하는 moneyTray 는 100 원을 실제 객체로 저장한다. 이때, moneyTray 는 연결 리스트 기반 stack 알고리즘을 따른다. moneyBox 는 추가적으로, 요구사항에 의하여, 잔돈을 계산하는 메서드를 가지고 있으며, 잔돈 단위(5 개) 별 개수가 저장된 배열을 반환한다. 또한, moneyLog()를 통하여 매출 기록을 산출한다.

추가적으로, PaymentMachine 과 MoneyBox 는 부모, 자식 관계인데, 이는 PaymentMachine 이 MoneyBox 의 모든 기능을 사용할 수 있어야 한다고 생각했기 때문이다.

- linked list interface

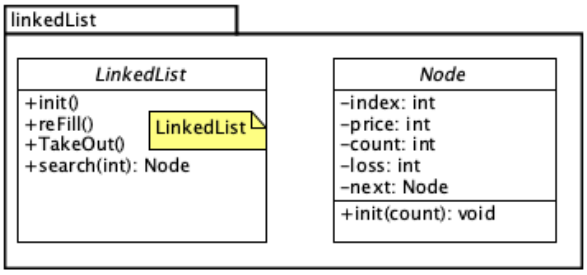


그림 5: 연결 리스트 인터페이스의 클래스 다이어그램

앞서 정의한 클래스 다이어그램으로부터, 연결 리스트가 매우 많이 사용된다는 것을 바탕으로, 연결 리스트의 일관성을 보장하기 위하여 연결 리스트와 노드를 인터페이스로 정의했다. 또한, 각 노드는 저장하는 정보가 다름으로, 제네릭을 사용할 전망이다.

- admin

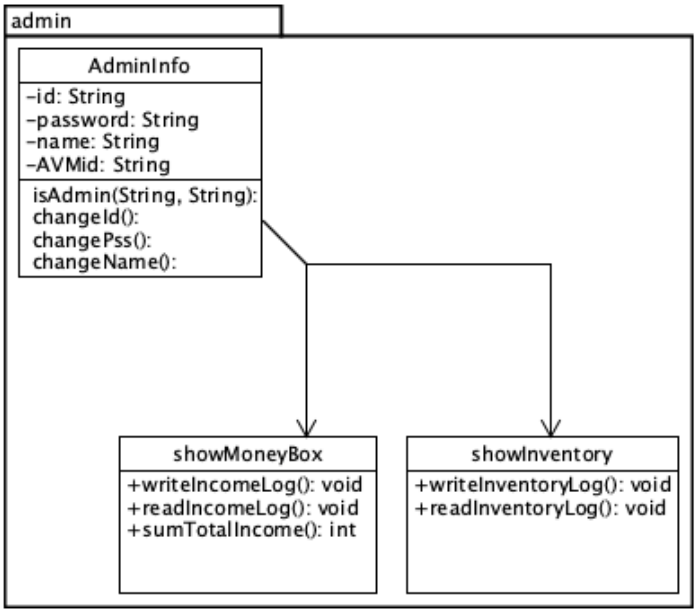


그림 6: admin 모듈의 클래스 다이어그램

Admin 모듈의 AdminInfo 클래스는 관리자의 모든 정보를 저장 및 관리한다. 관리자의 비밀번호를 입력받고 관리자 페이지 접근을 관리하기도 한다. 또한, showMoneyBox, showInventory는 로그를 산출하는 기능을 맡는다.

- 최종 관계

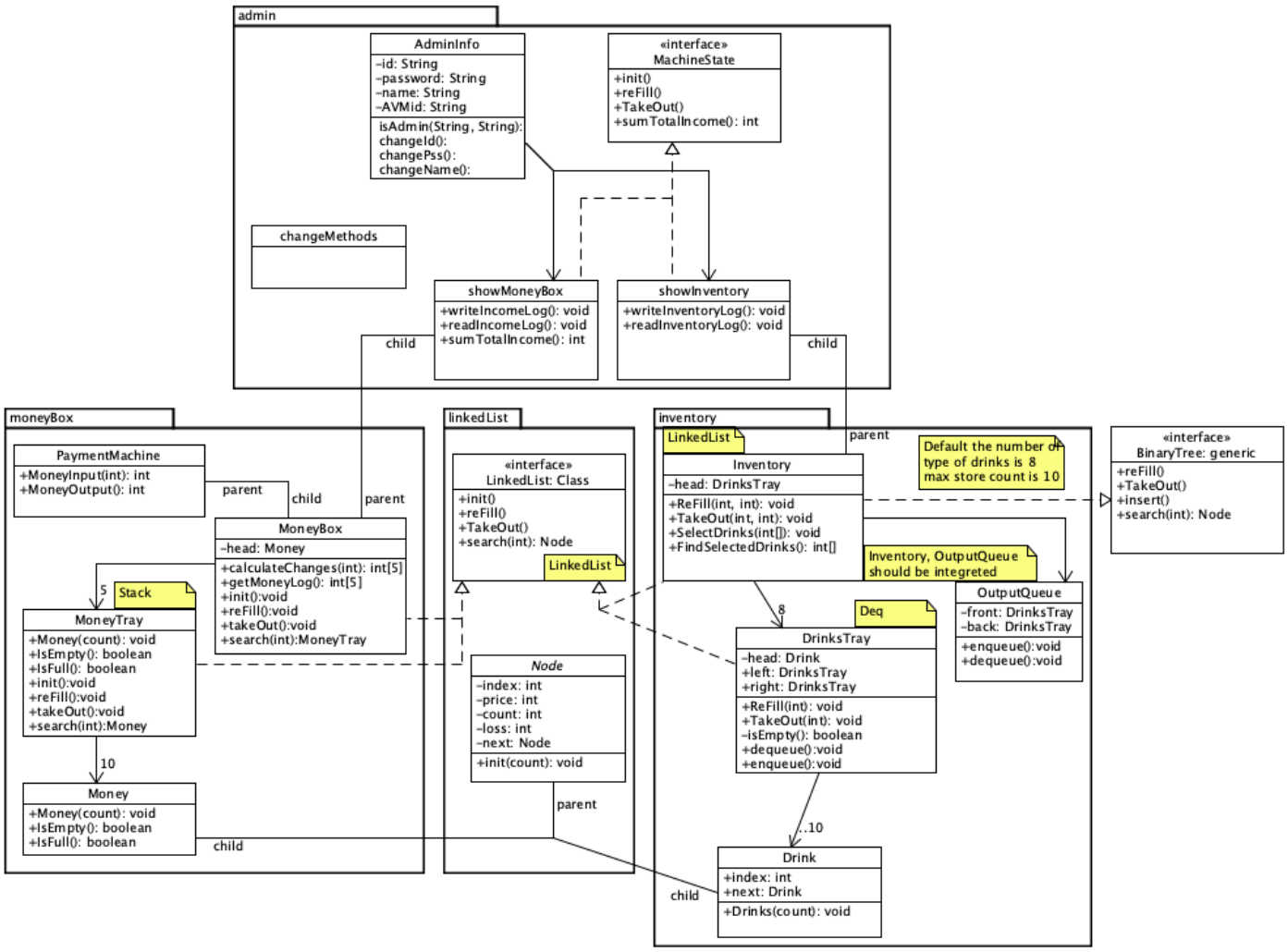


그림 7: 최종 클래스 다이어그램

3. 시퀀스 다이어그램을 통한 동작 순서관계 파악

앞서 진행한 클래스 다이어그램에 이어 시퀀스 다이어그램(sequence diagram)은 이름 그대로 연속적인 순서 관계를 나타내는 다이어그램이다. 객체들 간의 상호작용이 시간의 흐름에 따라 어떤 순서로 이루어지는지를 시각적으로 UML 로 표현을 할 수 있다.

다이어그램의 상단에는 액터나 객체가 가로로 나열되고, 각 객체 아래로 생명선이 점선으로 그려진다. 객체가 활성화되는 구간은 생명선 위에 직사각형 막대로 표시되며, 이때 메시지를 주고받는다. 메시지는 객체 사이를 연결하는 실선 화살표로 나타내며, 동기 메시지는 꽉 찬 화살표, 비동기 메시지는 빈 화살표로 표기한다. 응답 메시지는 점선 화살표로 표현된다. 반복이나 조건 분기는 프레임을 이용해 loop, opt 등으로 나타낼 수 있다. 객체가 소멸되는 시점은 생명선 끝에 X 표시로 나타난다. 시퀀스 다이어그램은 시스템의 동작 시나리오를 시간 순서에 따라 명확하게 보여주며, 객체 간 상호작용과 메시지 흐름을 이해하는 데 효과적이다.

1. 사용자 구매 단계

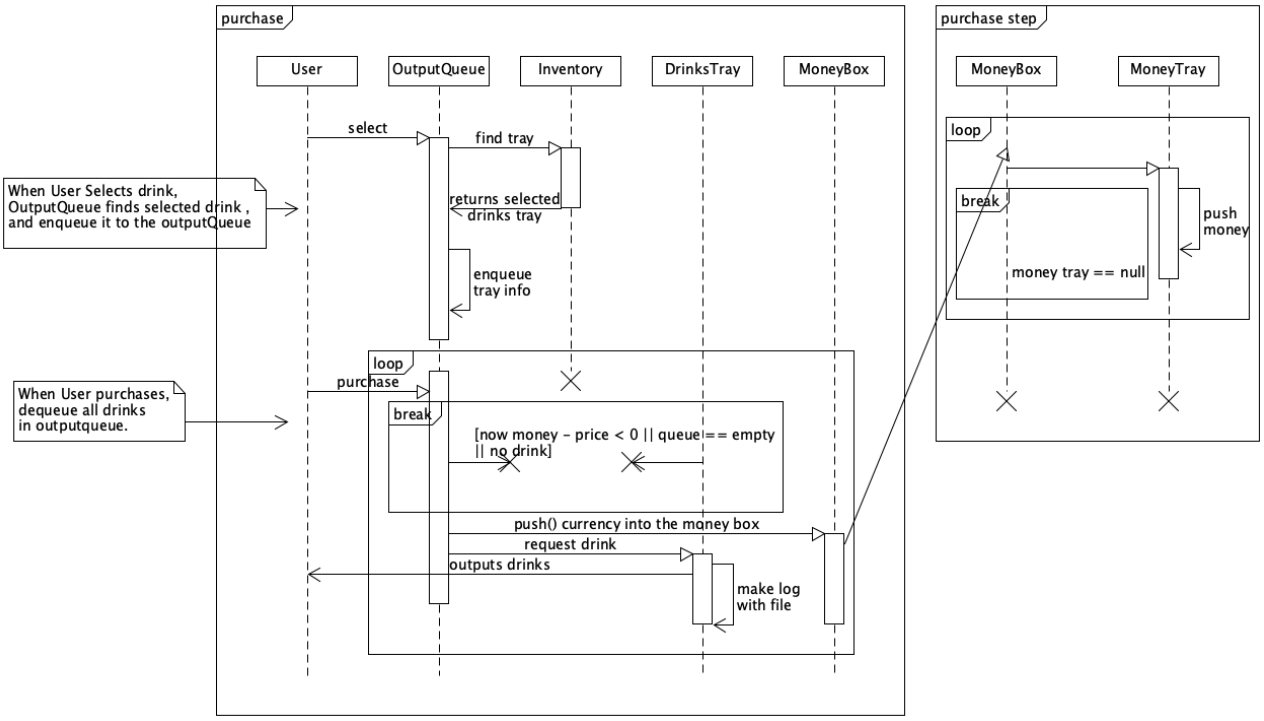


그림 8: 시퀀스 다이어그램을 통한 사용자 구매 패턴

- 음료 선택 단계

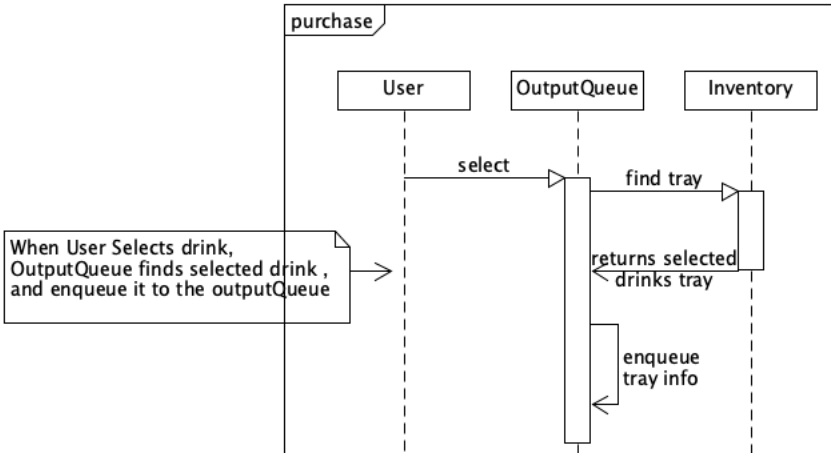


그림 9: 사용자의 음료 선택 단계

그림 8을 확대한 그림 9를 보면, 사용자가 음료를 선택하였을 때, 어떠한 상호작용이 발생하는 지를 나타내었다. OutputQueue는 사용자가 음료를 선택하면, Inventory로부터 하여금 사용자가 선택한 음료 종류를 보관하는 drinks tray를 찾아내고, 해당 트레이 위치를 enqueue한다.

- 음료 구매 단계

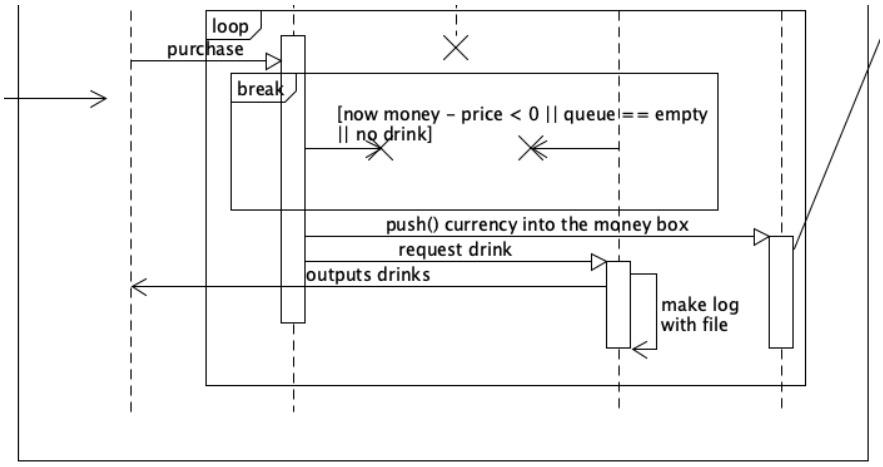


그림 10: 사용자 음료 구매 단계

위 사진은 그림 8 의 확대 사진으로, 사용자가 음료를 선택한 후, 구매 버튼을 눌렀을 때 작동하는 과정이다. OutputQueue 에 있는 음료를 모두 배출할 때 까지 또는 돈이 부족할 때 까지 반복하고, 정상적인 과정에서는 돈통에 돈을 보관하고, drinksTray 에 음료를 요청한다.

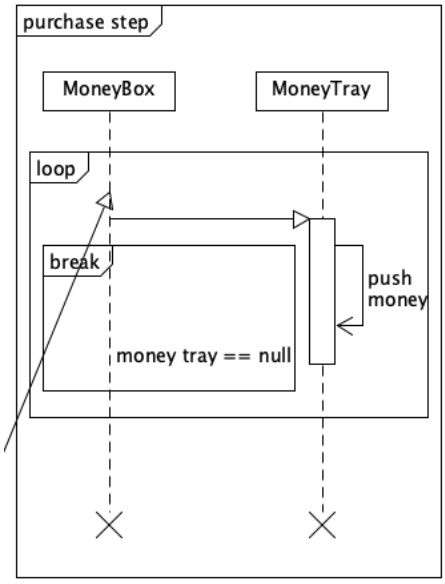


그림 11: 그림 8 의 확대사진으로, 그림 10 의 오른쪽 부분, 돈 저장 과정

위 사진은 사용자로부터 들어온 돈을 저장하는 과정으로, moneyBox 에서 화폐를 모두 넣을 때 까지 돈을 저장한다.

- 화폐 반환 과정

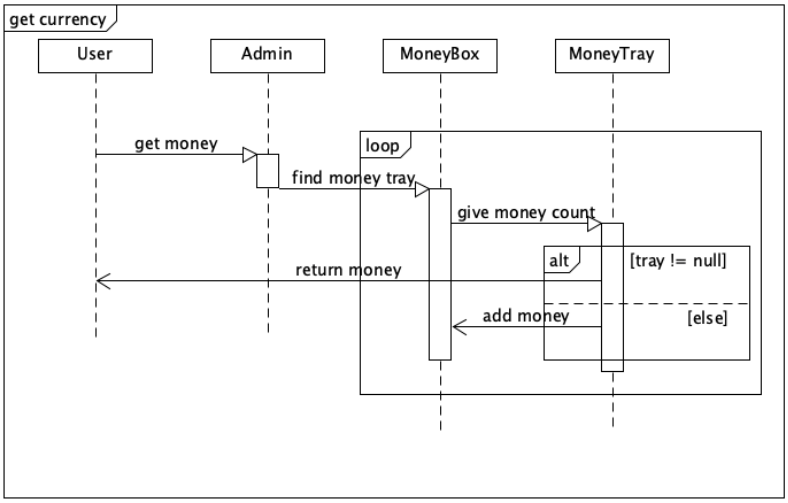
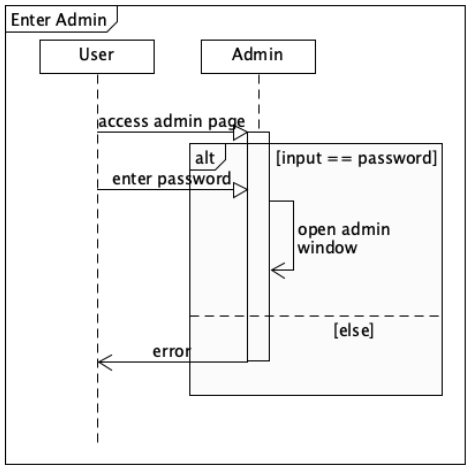


그림 12: 사용자로부터 받은 돈을 반환

위 사진은 사용자로부터 입력받은 돈을 모두 반환하는 과정으로, 사용자로부터 요청이 들어오면, 관리자 클래스에서 돈을 요청하고, moneyBox 에서 moneyTray 를 찾고 돈을 인출하는 과정을 반복한다.

- 기타



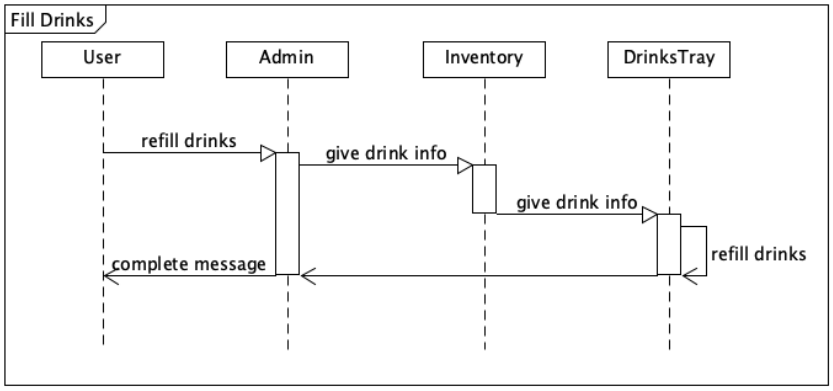
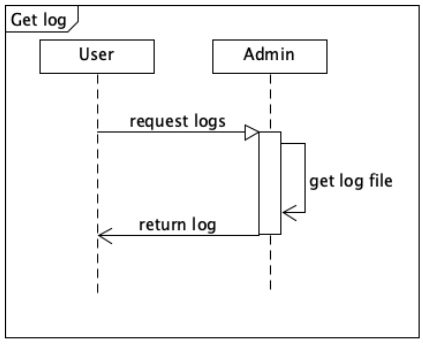


그림 13: 관리자 윈도우 접근(우), 로그 산출(우), 재고 보충(상)



JAVA 음료 자동 판매기(Auto Vending Machin)

결과 보고서

목차

1. 최종 구조

- 모듈 구성

최종 모듈 구성은 다음과 같다.

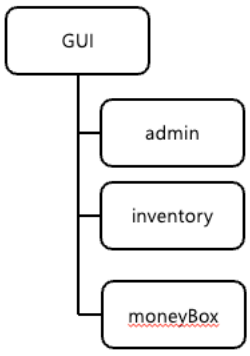


사진: 최종 모듈 구성

- 사용자 인터페이스 구성

최종 사용자 인터페이스의 모습은 다음과 같다.

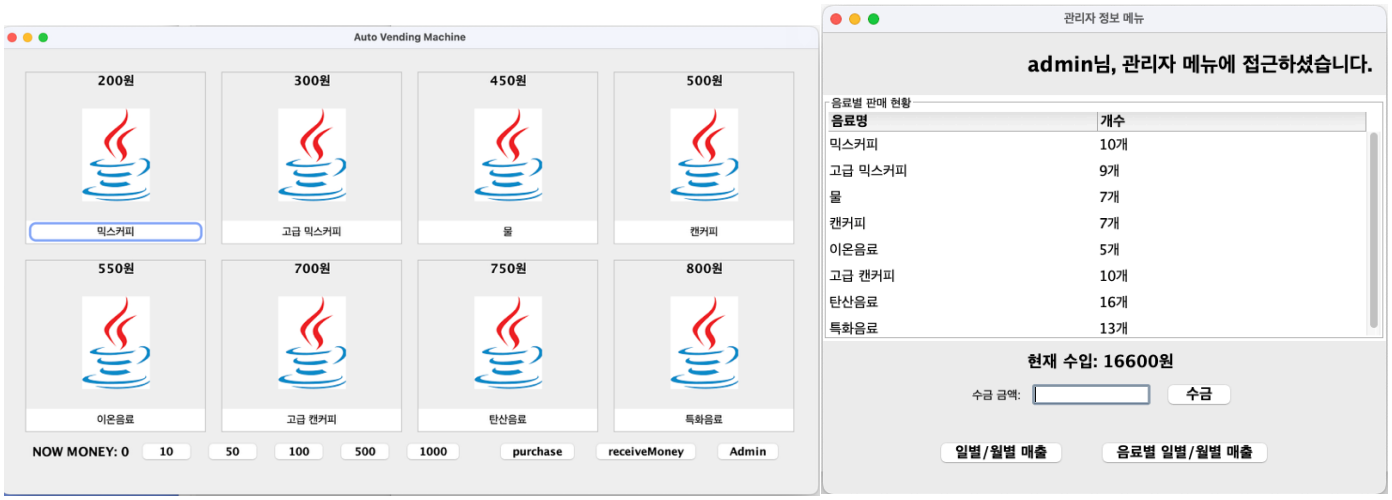


사진: 좌: 메인 윈도우, 우: 관리자 윈도우

- 요구 명세서 구현 결과

○ 기능 요구사항:

□ 판매하는 음료의 개수는 기본적으로 8개로 하되, 각각 200원(믹스커피), 300원(고급믹스커피), 450원(물), 500원(캔커피), 550원(이온음료), 700원(고급캔커피), 750원 (탄산음료), 800원 (특화음료)으로 한다.

□ 각각 음료의 재고는 기본적으로 10개로 하며, 10개 이상 음료가 배출되었을 때는 해당 음료 (물품)에 대해 품절 표시를 하고 해당 음료를 더 이상 선택할 수 없어야 한다 (단, 이전에 재고가 보충되었을 경우에는 해당 재고만큼의 판매가 가능해야 함).

□ 입력 받을 수 있는 화폐의 단위는 10원, 50원, 100원, 500원, 1,000원으로만 한정하되, 지폐로 입력 받을 수 있는 금액의 상한선은 5,000원까지이며, 지폐와 동전을 모두 포함하여 총 7,000원을 초과하여 입력 받을 수 없다 (화폐의 입력 금액에 따라 판매 가능 음료가 표기되어야 한다). 이 때, 화폐를 입력 받는 변수는 반드시 동적 할당을 사용해야 하며, 화폐가 반환되거나 음료의 판매가 끝나면 동적 할당을 해제해야 한다.

□ 기본적으로 거스름 돈을 가지고 있어야 하며 (각 동전별로 10개를 기본 값으로 하고 음료의 개수와 마찬가지로 생성자를 사용하여 초기화 할 것), 거스름 돈 반환 혹은 음료 판매에 따른 동전의 가감이 구현되어야 한다. 또한 음료 자판기에서 화폐를 입력 받을 때는 반드시 사용자 요구에 의한 화폐 반환이 가능해야 한다. 단, 거스름돈이 없는 경우, 거스름돈 없음이란 표기를 한 후 이에 맞는 화폐 입력을 받아야 한다.

□ 음료가 배출된 이후에도 반드시 다시 화폐를 입력 받을 수 있는 상태가 되어야 한다.

□ 관리자 메뉴 (비밀번호 확인을 통해 관리자 메뉴에 접근할 수 있도록 작성하되, 비밀번호는 반드시 특수문자 및 숫자가 각각 하나 이상 포함된 8자리 이상으로 설정할 수 있도록 해야 하며 필요시 언제든 변경 가능해야 함)는 다음과 같은 조건을 만족해야 한다.

- 일별/월별 매출 산출, 각 음료의 일별/월별 매출, 각 음료의 재고 보충 (재고 보충은 오프라인 형태)이 가능해야 한다. 이 때, 일별/월별 매출은 사전에 파일로 저장해 놓

은 파일(각자 만들어 사용)을 사용하여야 하며, 현재의 자판기 매출은 사전에 저장해 놓은 파일과 연관성을 가지고 있어야 한다.

- 관리자 메뉴에서는 현재 자판기 내의 화폐현황을 손쉽게 파악할 수 있어야 하며, 관리자가 "수금"이란 메뉴를 선택할 경우, 해당 금액을 수금할 수 있어야 한다. 단, 이 경우에도 반환을 위한 최소한의 화폐 (임의로 지정할 것)는 남겨두어야 한다.
- 관리자 메뉴에서는 각 음료의 판매가격, 판매이름을 사용자의 입력을 통해 언제든 바꿀 수 있어야 한다.
- 관리자 메뉴와 관련된 모든 사항들은 파일로 읽기/쓰기가 되어야 한다. (최소 기록 사항: 일별/월별 매출, 재고 소진 날짜 혹은 주기)

□ 관리자 메뉴가 활성화 되어 있는 경우, 화면이 관리자 전용 메뉴로 변경 되어야 하며, 관리자 메뉴가 활성화 되어 있는 경우에는 다른 기능들은 동작하지 않아야 한다. 즉, 관리자 화면과 판매 화면은 각각 독립적으로 동작해야 한다.

□ 반드시 예외 처리 및 주석 (프로그램, 함수, 라인 주석 등)이 반드시 포함되어야 한다.

□ 예외처리시 사용하는 언어의 예외처리 방식에 따른 동작을 사용해야 하며, 모든 부분에 예외처리가 잘 이루어져야 한다.

□ 상기한 바와 같은 프로그램의 실행은 모두 GUI 환경에서 동작하여야 한다.

□ 사용하는 모든 자료구조는 직접 만들어서 사용해야 한다.

□ 각 음료의 재고 수량은 Linked-List를 통해 구현되어야 한다.

□ 본 프로그램의 작성 시 Stack과 Queue가 적절한 곳에 각 1회 이상씩 사용되어야 한다.

□ 본 프로그램의 작성 시 Tree 구조가 적절한 곳에 1회 이상씩 사용되어야 한다.

□ 본 프로그램의 작성 시 다양한 Sort 및 Search 기법이 적절한 곳에 각 1회 이상씩 사용되어야 한다.

2. 요구 사항별 소스 코드

□ 판매하는 음료의 개수는 기본적으로 8개로 하되, 각각 200원(믹스커피), 300원(고급믹스커피), 450원(물), 500원(캔커피), 550원(이온음료), 700원(고급캔커피), 750원 (탄산음료), 800원 (특화음료)으로 한다.

Package: inventory, Module: Inventory

```
import linkedList.BinaryTree;

/**
 * class name: InventoryTree
 * latest modify date: 2025.05.18
 * run environment: MacOS 15.4.1(24E263)
 *
 * Feature: Implemented class that resembles Drinks Tray storing drinks in reality
 *
 * @author Sunghun Wang
 * @version 0.9, implemented class
 * @see
 */

public class Inventory implements BinaryTree<DrinksTray> {

    DrinksTray head;

    public Inventory(){
        head = new DrinksTray(8, 0);
```

```
        reFill(new DrinksTray(1, 200));
        reFill(new DrinksTray(3, 300));
        reFill(new DrinksTray(5, 450));
        reFill(new DrinksTray(7, 500));
        reFill(new DrinksTray(9, 550));
        reFill(new DrinksTray(11, 700));
        reFill(new DrinksTray(13, 750));
        reFill(new DrinksTray(15, 800));
    }
    public void reFill(DrinksTray element){
        DrinksTray tmp = head;
        while(true){
            if (element.trayNumber < tmp.trayNumber) {
                if(tmp.left == null){
                    tmp.left = element;
//                  System.out.println("left");
                    break;
                }else{
                    tmp = tmp.left;
//                  System.out.println("left");
                }
            } else if (element.trayNumber > tmp.trayNumber) {
                if(tmp.right == null){
                    tmp.right = element;
//                  System.out.println("right");
                    break;
                }else{
                    tmp = tmp.right;
//                  System.out.println("right");
                }
            }
        }
    }
    public DrinksTray takeOut(int id, int count){
        DrinksTray tmp = new DrinksTray(1, 0);
        return tmp;
    }
    public void insert(DrinksTray element){

    }
    public DrinksTray search(int id){
        DrinksTray tmp = this.head;

        while(tmp.trayNumber != id){
            if(id < tmp.trayNumber && tmp.left != null){
                tmp = tmp.left;
            }else if(id > tmp.trayNumber && tmp.right != null) {
                tmp = tmp.right;
            }else{
                break;
            }
        }
        return tmp;
    }
    public void del(int id){

    }
}
```

Package: inventory, Module: DrinksTray

```
import linkedList.BinaryTree;

//implement interface which is generic
public class Inventory implements BinaryTree<DrinksTray> {

    DrinksTray head;

    //generate inventory tree
    public Inventory(){
        head = new DrinksTray(8, 0);
        reFill(new DrinksTray(1, 200));
        reFill(new DrinksTray(3, 300));
        reFill(new DrinksTray(5, 450));
        reFill(new DrinksTray(7, 500));
        reFill(new DrinksTray(9, 550));
        reFill(new DrinksTray(11, 700));
        reFill(new DrinksTray(13, 750));
        reFill(new DrinksTray(15, 800));
    }

    //add drinks Trays
    public void reFill(DrinksTray element){
        DrinksTray tmp = head;
        while(true){
            if (element.trayNumber < tmp.trayNumber) {
                if(tmp.left == null){
                    tmp.left = element;
//                    System.out.println("left");
                    break;
                }else{
                    tmp = tmp.left;
//                    System.out.println("left");
                }
            } else if (element.trayNumber > tmp.trayNumber) {
                if(tmp.right == null){
                    tmp.right = element;
//                    System.out.println("right");
                    break;
                }else{
                    tmp = tmp.right;
//                    System.out.println("right");
                }
            }
        }
    }

    public DrinksTray takeOut(int id, int count){
        DrinksTray tmp = new DrinksTray(1, 0);
        return tmp;
    }
    public void insert(DrinksTray element){

    }

    //binary search
    public DrinksTray search(int id){
        DrinksTray tmp = this.head;

        while(tmp.trayNumber != id){ // id가 같은 노드 찾을 때 까지 탐색
            if(id < tmp.trayNumber && tmp.left != null){    // 왼쪽이 비어있지 않고 현재
id가 목표보다 클 때
```

```
                tmp = tmp.left;
            }else if(id > tmp.trayNumber && tmp.right != null) {//오른쪽이 비어있지 않고
현재 id가 목표보다 작을 때
                tmp = tmp.right;
            }else{
                break;
            }
        }
        return tmp; // 못찾으면 null
    }
    public void del(int id){

    }
}
```

□ 각각 음료의 재고는 기본적으로 10개로 하며, 10개 이상 음료가 배출되었을 때는 해당 음료 (물품)에 대해 품절 표시를 하고 해당 음료를 더 이상 선택할 수 없어야 한다 (단, 이전에 재고가 보충되었을 경우에는 해당 재고만큼의 판매가 가능해야 함).

Module: AutoVendingMachine

```
…생략
Stack<Integer> disabledDrinks = new Stack<>();  // 품절 음료 저장

// 큐가 비어있지 않으면 알림 시작
if (!outputQueue.isEmpty()) outputQueue.alert();

// 큐 처리 루프
while (!outputQueue.isEmpty()) {
    int nowMoney = outputQueue.getTotalMoneyInput();
    int price = outputQueue.front.tray.price;

    // 잔액 검증
    if (nowMoney >= price) {
        int result = outputQueue.takeOut_(1);  // 음료 배출 시도

        // 품절 음료 처리 (0: 성공, 0!=: 품절 ID)
        if (result != 0) {
            disabledDrinks.push(result);
            continue;
        }

        // 금액 업데이트
        outputQueue.subTotalMoneyInput(price);
        moneyBox.changeNstore(price);  // 금고에 금액 저장
    } else {
        // 잔액 부족 처리
        outputQueue.takeOut(0);  // 음료 제거
        noMoney.setVisible(true);  // 경고 메시지 표시

        // 3초 후 메시지 숨김
        new javax.swing.Timer(3000, evt -> {
            noMoney.setVisible(false);
            ((javax.swing.Timer) evt.getSource()).stop();
        }).start();
        break;
    }
}

// 음료 배출 창 위치 초기화
```

20243522 왕성훈

```
DrinkOut.baseX = 300;
DrinkOut.baseY = 300;

// 품절 음료 버튼 비활성화
while (!disabledDrinks.empty()) {
    int id = disabledDrinks.pop();
    // ID에 해당하는 버튼 찾아 비활성화
    for (Select button : buttons) {
        if (button.drinkId != null && button.drinkId == id) {
            button.btn.setEnabled(false);
            break;
        }
    }
}
… 생략
```

1 입력 받을 수 있는 화폐의 단위는 10원, 50원, 100원, 500원, 1,000원으로만 한정하되, 지폐로 입력 받을 수 있는 금액의 상한선은 5,000원까지이며, 지폐와 동전을 모두 포함하여 총 7,000원을 초과하여 입력 받을 수 없다 (화폐의 입력 금액에 따라 판매 가

Module: AutoVendingMachine

```
…생략
class MoneyInput implements ActionListener {
    int MAX_MONEY = 7000;  // 최대 투입 가능 금액
    OutputQueue outputQueue;  // 출력 큐 참조
    MoneyBox moneyBox;  // 금고 참조
    JLabel display;  // 금액 표시 라벨

    MoneyInput(OutputQueue outputQueue, MoneyBox moneyBox, JLabel display) {
        this.outputQueue = outputQueue;
        this.moneyBox = moneyBox;
        this.display = display;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        JButton btn = (JButton) e.getSource();
        int amount = Integer.parseInt(btn.getText());  // 버튼 텍스트에서 금액 파싱
        int currentMoney = outputQueue.getTotalMoneyInput();

        // 최대 금액 검증 후 투입 금액 추가
        if (currentMoney + amount <= MAX_MONEY) {
            outputQueue.sumTotalMoneyInput(amount);
            display.setText("NOW MONEY: " + outputQueue.getTotalMoneyInput());
        }
    }
}
…생략
```

1 기본적으로 거스름 돈을 가지고 있어야 하며 (각 동전별로 10개를 기본 값으로 하고 음료의 개수와 마찬가지로 생성자를 사용하여 초기화 할 것), 거스름 돈 반환 혹은 음료 판매에 따른 동전의 가감이 구현되어야 한다. 또한 음료 자판기에서 화폐를 입력 받을 때는 반드시 사용자 요구에 의한 화폐 반환이 가능해야 한다. 단, 거스름돈이 없는 경우, 거스름돈 없음이란 표기를 한 후 이에 맞는 화폐 입력을 받아야 한다.

Package: moneybox, Module: MoneyTray

```
…생략
```

```
public class MoneyTray implements LinkedList {
    //stack
    Money top;
    //money box linked list
    MoneyTray next;
    //10, 50, 100, 500, 1000
    int moneyUnit;
    //maximum capacity of money tray
    final int sizeOfMoneyTray = 50;
    //sum total income
    static int totalIncome = 0;

    MoneyTray(int moneyUnit){
        this.next = null;
        //moneyUnit is price of the currency
        this.top = null;
        this.moneyUnit = moneyUnit;
        init();
    }
    //head
    //initialize money tray (stack)
    @Override
    public void init() {
        //add 10
        this.reFill(10);
    }
…생략
```

위 코드는 초기화될 때 화폐를 10 개로 세팅한다.

Package: moneybox, Module: moneyTray

```
…생략
/**
 * 수금 처리 (판매 금액 저장)
 * @param price 저장할 금액
 */
public void changeNstore(int price) {
    MoneyTray current = head;
    int[] units = {1000, 500, 100, 50, 10};  // 화폐 단위 배열

    for (int unit : units) {
        int count = price / unit;  // 해당 단위 개수 계산
        price %= unit;             // 남은 금액 갱신

        // 해당 단위 트레이에 화폐 추가
        if (current != null) {
            current.reFill(count);
            totalMoneyInMachine += count * unit;  // 총액 업데이트
            current = current.next;
        }
    }
}
/**
 * 거스름돈 반환 처리
 * @param price 반환할 금액
 */
public void changeNreceive(int price) {
    MoneyTray current = head;
    int[] units = {1000, 500, 100, 50, 10};
    MoneyOut out = new MoneyOut();  // 화폐 출력 핸들러
```

```
    for (int unit : units) {
        int count = price / unit;
        price %= unit;

        if (current != null) {
            // 트레이에서 화폐 인출 시도
            int shortfall = current.takeOut_(count);

            // 성공 시
            if (shortfall == 0) {
                totalMoneyInMachine -= count * unit;
                for (int j = 0; j < count; j++) {
                    out.alert(unit);  // 화폐 출력
                }
            }
            // 실패 시 (재고 부족)
            else {
                int available = count - shortfall;
                // 가능한 만큼 인출
                for (int j = 0; j < available; j++) {
                    out.alert(unit);
                }
                out.alert(-1);  // 오류 알림
                totalMoneyInMachine -= available * unit;
            }
            current = current.next;
        }
    }
}
…생략
```

관리자 페이지 코드

```
public class AdminInfo extends JFrame implements ActionListener {
    private static final String CREDENTIALS_FILE = "admin_credentials.txt";
    private JButton accessAdminBtn;
    private JButton changePasswordBtn;
    private static String currentId;
    private static String currentPassword;

    private int currentIncome = 0; // 수입 저장 변수
    private JLabel incomeLabel; // 수입 라벨 참조 변수
    private int totalMoneyInMachine;
    private JLabel totalMoney;
    JButton[] allbtns;

    OutputQueue outputQueue;
    MoneyBox moneyBox;

    public AdminInfo(OutputQueue outputQueue,
                     MoneyBox moneyBox, JButton[] allbtns) {
        super("관리자 정보 시스템");

        this.outputQueue = outputQueue;
        this.moneyBox = moneyBox;

        loadCredentials(); // 파일에서 아이디와 비밀번호 불러오기

        // 버튼 설정
        accessAdminBtn = new JButton("관리자 메뉴 접근");
        changePasswordBtn = new JButton("비밀번호 변경");
```

```
        this.allbtns = allbtns;

        accessAdminBtn.addActionListener(this);
        changePasswordBtn.addActionListener(this);

        this.totalMoneyInMachine = moneyBox.totalMoneyInMachine;

        // 레이아웃 설정
        JPanel panel = new JPanel(new GridLayout(2, 1, 10, 10));
        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
        panel.add(accessAdminBtn);
        panel.add(changePasswordBtn);

        add(panel);
        setSize(350, 200);
        setLocationRelativeTo(null);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == accessAdminBtn) {
            authenticateAdmin();
        } else if (e.getSource() == changePasswordBtn) {
            changePassword();
        }
    }

    // 인증 처리 (아이디, 비밀번호 모두 입력)
    private void authenticateAdmin() {
        JPanel panel = new JPanel(new GridLayout(2, 2, 5, 5));
        JTextField idField = new JTextField(15);
        JPasswordField pwField = new JPasswordField(15);

        panel.add(new JLabel("아이디:"));
        panel.add(idField);
        panel.add(new JLabel("비밀번호:"));
        panel.add(pwField);

        int result = JOptionPane.showConfirmDialog(
                this, panel, "관리자 인증",
                JOptionPane.OK_CANCEL_OPTION
        );

        if (result == JOptionPane.OK_OPTION) {
            String inputId = idField.getText();
            String inputPw = new String(pwField.getPassword());
            if (inputId.equals(currentId) && inputPw.equals(currentPassword)) {
                openAdminMenu();
            } else {
                JOptionPane.showMessageDialog(this,
                        "아이디 또는 비밀번호가 일치하지 않습니다.", "인증 실패",
                        JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    // 비밀번호 변경 (관리자 페이지 내에서 변경 가능)
    private void changePassword() {
        // 1. 현재 아이디/비밀번호 확인
        JPanel authPanel = new JPanel(new GridLayout(2, 2, 5, 5));
        JTextField idField = new JTextField(15);
```

```
        JPasswordField pwField = new JPasswordField(15);

        authPanel.add(new JLabel("아이디:"));
        authPanel.add(idField);
        authPanel.add(new JLabel("비밀번호:"));
        authPanel.add(pwField);

        int authResult = JOptionPane.showConfirmDialog(
                this, authPanel, "관리자 인증",
                JOptionPane.OK_CANCEL_OPTION
        );

        if (authResult != JOptionPane.OK_OPTION) return;

        String inputId = idField.getText();
        String inputPw = new String(pwField.getPassword());
        if (!inputId.equals(currentId) || !inputPw.equals(currentPassword)) {
            JOptionPane.showMessageDialog(this,
                    "아이디 또는 비밀번호가 일치하지 않습니다.", "오류",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }

        // 2. 새 비밀번호 입력 및 유효성 검사
        JPanel changePanel = new JPanel(new GridLayout(3, 2, 5, 5));
        JPasswordField newPwField = new JPasswordField(15);
        JPasswordField confirmPwField = new JPasswordField(15);

        changePanel.add(new JLabel("새 비밀번호:"));
        changePanel.add(newPwField);
        changePanel.add(new JLabel("비밀번호 확인:"));
        changePanel.add(confirmPwField);

        int changeResult = JOptionPane.showConfirmDialog(
                this, changePanel, "비밀번호 변경",
                JOptionPane.OK_CANCEL_OPTION
        );

        if (changeResult == JOptionPane.OK_OPTION) {
            String newPw = new String(newPwField.getPassword());
            String confirmPw = new String(confirmPwField.getPassword());

            // 새 비밀번호 유효성 검사
            if (!isValidPassword(newPw)) {
                JOptionPane.showMessageDialog(this,
                        "비밀번호는 8 자리 이상이며, 숫자와 특수문자를 각각 1 개 이상 포함해야
합니다.",
                        "유효성 검사 실패", JOptionPane.WARNING_MESSAGE);
                return;
            }

            // 비밀번호 확인
            if (!newPw.equals(confirmPw)) {
                JOptionPane.showMessageDialog(this,
                        "새 비밀번호가 일치하지 않습니다.", "오류",
                        JOptionPane.ERROR_MESSAGE);
                return;
            }

            // 비밀번호 업데이트
            currentPassword = newPw;
```

```
            saveCredentials();
            JOptionPane.showMessageDialog(this,
                    "비밀번호가 성공적으로 변경되었습니다.", "완료",
                    JOptionPane.INFORMATION_MESSAGE);
        }
    }

    // 비밀번호 유효성 검사 (숫자+특수문자 포함 8 자리 이상)
    private boolean isValidPassword(String password) {
        if (password.length() < 8) return false;

        boolean hasDigit = false;
        boolean hasSpecial = false;
        String specialChars = "!@#$%^&*()-_=+[]{}|;:'\",.<>/?`~";

        for (char c : password.toCharArray()) {
            if (Character.isDigit(c)) hasDigit = true;
            if (specialChars.indexOf(c) >= 0) hasSpecial = true;
        }
        return hasDigit && hasSpecial;
    }

    private void openAdminMenu() {
        JFrame adminFrame = new JFrame("관리자 정보 메뉴");
        adminFrame.setSize(600, 520);
        adminFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        adminFrame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                for(int i = 0; i<16; i++){
                   allbtns[i].setEnabled(true);

                }
            }
        });

        // 환영 메시지
        JLabel welcomeLabel = new JLabel(currentId + "님, 관리자 메뉴에 접근하셨습니다.",
SwingConstants.CENTER);
        welcomeLabel.setFont(new Font("맑은 고딕", Font.BOLD, 22));
        welcomeLabel.setBorder(BorderFactory.createEmptyBorder(20, 0, 10, 0));

        // 음료 종류별 개수 표
        String[] columns = {"음료명", "개수"};
        String[] beverageTypes = {"믹스커피", "고급 믹스커피", "물", "캔커피", "이온음료",
"고급 캔커피", "탄산음료", "특화음료"};
        Map<String, Integer> beverageCounts = getBeverageCounts();
        Object[][] data = new Object[beverageTypes.length][2];
        for (int i = 0; i < beverageTypes.length; i++) {
            data[i][0] = beverageTypes[i];
            data[i][1] = beverageCounts.getOrDefault(beverageTypes[i], 0) + "개";
        }
        JTable table = new JTable(data, columns);
        table.setFont(new Font("맑은 고딕", Font.PLAIN, 15));
        table.setRowHeight(28);
        table.setEnabled(false);
        table.getTableHeader().setFont(new Font("맑은 고딕", Font.BOLD, 15));
        table.setShowGrid(false);

        JScrollPane tablePane = new JScrollPane(table);
        tablePane.setBorder(BorderFactory.createTitledBorder("음료별 판매 현황"));
```

```
        tablePane.setPreferredSize(new Dimension(320, 250));

        // 현재 수입 라벨
        incomeLabel = new JLabel("현재 수입: " + moneyBox.totalMoneyInMachine + "원",
SwingConstants.CENTER);
        incomeLabel.setFont(new Font("맑은 고딕", Font.BOLD, 18));
        incomeLabel.setBorder(BorderFactory.createEmptyBorder(10, 0, 10, 0));

        // 수금 입력 패널
        JPanel collectPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 0));
        JLabel amountLabel = new JLabel("수금 금액:");
        JTextField amountField = new JTextField(10);
        JButton collectBtn = new JButton("수금");

        collectBtn.setFont(new Font("맑은 고딕", Font.BOLD, 16));
        collectBtn.setPreferredSize(new Dimension(80, 30));

        // 수금 버튼 액션
        collectBtn.addActionListener(e -> {
            try {
                int amount = Integer.parseInt(amountField.getText().trim());
                currentIncome += amount;
                moneyBox.changeNreceive(amount);
                incomeLabel.setText("현재 수입: " + moneyBox.totalMoneyInMachine + "원");
                saveIncomeData();
                amountField.setText("");
                JOptionPane.showMessageDialog(adminFrame, amount + "원이 수금되었습니다.");
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(adminFrame, "유효한 숫자를 입력하세요!", "입력
오류", JOptionPane.ERROR_MESSAGE);
            }
        });

        collectPanel.add(amountLabel);
        collectPanel.add(amountField);
        collectPanel.add(collectBtn);

        // 버튼 패널
        JButton logViewBtn = new JButton("일별/월별 매출");
        JButton logViewBtn2 = new JButton("음료별 일별/월별 매출");
        logViewBtn.setFont(new Font("맑은 고딕", Font.BOLD, 16));
        logViewBtn2.setFont(new Font("맑은 고딕", Font.BOLD, 16));

        JPanel btnPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 30, 0));
        btnPanel.setBorder(BorderFactory.createEmptyBorder(10, 0, 15, 0));
        btnPanel.add(logViewBtn);
        btnPanel.add(logViewBtn2);

        // 로그 보기 기능 연결
        logViewBtn.addActionListener(e -> showSalesSummary());
        logViewBtn2.addActionListener(e -> showBeverageSales());

        // 전체 레이아웃 (BoxLayout)
        JPanel content = new JPanel();
        content.setLayout(new BoxLayout(content, BoxLayout.Y_AXIS));
        content.add(welcomeLabel);
        content.add(Box.createVerticalStrut(10));
        content.add(tablePane);
        content.add(incomeLabel);      // 수입 라벨 (수금 버튼 위)
        content.add(collectPanel);     // 수금 입력 패널
```

```
        content.add(Box.createVerticalStrut(10));
        content.add(btnPanel);

        adminFrame.setContentPane(content);
        adminFrame.setLocationRelativeTo(null);
        adminFrame.setVisible(true);
    }

    // 수입 데이터 저장
    private void saveIncomeData() {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter("income.txt")))
{
            writer.write(String.valueOf(currentIncome));
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "수입 데이터 저장 실패: " +
e.getMessage());
        }
    }

    // 프로그램 시작 시 수입 데이터 로드
    private void loadIncomeData() {
        try (BufferedReader reader = new BufferedReader(new FileReader("income.txt")))
{
            currentIncome = Integer.parseInt(reader.readLine());
        } catch (IOException | NumberFormatException e) {
            currentIncome = 0;
        }
    }
    // 음료 종류별 개수 계산
    private Map<String, Integer> getBeverageCounts() {
        Map<String, Integer> counts = new HashMap<>();
        // 음료 종류 초기화
        String[] beverages = {"믹스커피", "고급 믹스커피", "물", "캔커피", "이온음료", "고급
캔커피", "탄산음료", "특화음료"};
        for (String beverage : beverages) {
            counts.put(beverage, 0);
        }

        // 파일에서 데이터 읽기
        try (BufferedReader reader = new BufferedReader(new
FileReader("drinkLog.txt"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split("\t");
                if (parts.length < 2) continue;

                String beverage = parts[1].trim();
                if (counts.containsKey(beverage)) {
                    counts.put(beverage, counts.get(beverage) + 1);
                }
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "음료 데이터를 읽을 수 없습니다: " +
e.getMessage());
        }
        return counts;
    }

    // 음료 개수 표시용 라벨 생성 (HTML 형식)
    private JLabel createBeverageLabel(Map<String, Integer> counts) {
        StringBuilder html = new StringBuilder("<html><div style='text-
align:center;'>음료 개수:<br>");
```

```
        // 음료 종류별로 개수 표시
        for (Map.Entry<String, Integer> entry : counts.entrySet()) {
            html.append(entry.getKey()).append(":
").append(entry.getValue()).append("개<br>");
        }
        html.append("</div></html>");

        JLabel label = new JLabel(html.toString());
        label.setFont(new Font("맑은 고딕", Font.PLAIN, 14));
        return label;
    }

    // 일별/월별 매출 요약 보기
    private void showSalesSummary() {
        try {
            Map<String, Integer> dailySales = new TreeMap<>();
            Map<String, Integer> monthlySales = new TreeMap<>();

            processSalesData(dailySales, monthlySales);

            // 테이블 데이터 생성
            String[] columns = {"기간", "매출 건수"};
            DefaultTableModel dailyModel = new DefaultTableModel(columns, 0);
            for (Map.Entry<String, Integer> entry : dailySales.entrySet()) {
                dailyModel.addRow(new Object[]{entry.getKey(), entry.getValue()});
            }

            DefaultTableModel monthlyModel = new DefaultTableModel(columns, 0);
            for (Map.Entry<String, Integer> entry : monthlySales.entrySet()) {
                monthlyModel.addRow(new Object[]{entry.getKey(), entry.getValue()});
            }

            // 탭 패널 생성
            JTabbedPane tabbedPane = new JTabbedPane();

            JTable dailyTable = new JTable(dailyModel);
            tabbedPane.addTab("일별 매출", new JScrollPane(dailyTable));

            JTable monthlyTable = new JTable(monthlyModel);
            tabbedPane.addTab("월별 매출", new JScrollPane(monthlyTable));

            JFrame frame = new JFrame("매출 요약");
            frame.add(tabbedPane);
            frame.setSize(500, 400);
            frame.setVisible(true);

        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "데이터 파일을 읽을 수 없습니다: " +
e.getMessage());
        }
    }

    // 음료 종류별 매출 보기
    private void showBeverageSales() {
        try {
            Map<String, Map<String, Integer>> dailyBeverageSales = new TreeMap<>();
            Map<String, Map<String, Integer>> monthlyBeverageSales = new TreeMap<>();

            processBeverageData(dailyBeverageSales, monthlyBeverageSales);
```

```
            // 테이블 데이터 생성
            String[] columns = {"날짜", "음료", "판매량"};
            DefaultTableModel dailyModel = new DefaultTableModel(columns, 0);
            for (String date : dailyBeverageSales.keySet()) {
                for (Map.Entry<String, Integer> entry :
dailyBeverageSales.get(date).entrySet()) {
                    dailyModel.addRow(new Object[]{date, entry.getKey(),
entry.getValue()});
                }
            }

            String[] monthlyColumns = {"월", "음료", "판매량"};
            DefaultTableModel monthlyModel = new DefaultTableModel(monthlyColumns, 0);
            for (String month : monthlyBeverageSales.keySet()) {
                for (Map.Entry<String, Integer> entry :
monthlyBeverageSales.get(month).entrySet()) {
                    monthlyModel.addRow(new Object[]{month, entry.getKey(),
entry.getValue()});
                }
            }

            // 탭 패널 생성
            JTabbedPane tabbedPane = new JTabbedPane();

            JTable dailyTable = new JTable(dailyModel);
            tabbedPane.addTab("음료별 일별 판매량", new JScrollPane(dailyTable));

            JTable monthlyTable = new JTable(monthlyModel);
            tabbedPane.addTab("음료별 월별 판매량", new JScrollPane(monthlyTable));

            JFrame frame = new JFrame("음료별 판매 현황");
            frame.add(tabbedPane);
            frame.setSize(600, 400);
            frame.setVisible(true);

        } catch (IOException e) {
            JOptionPane.showMessageDialog(this, "데이터 파일을 읽을 수 없습니다: " +
e.getMessage());
        }
    }

    // 매출 데이터 처리
    private void processSalesData(Map<String, Integer> dailySales,
                                  Map<String, Integer> monthlySales) throws IOException {
        try (BufferedReader reader = new BufferedReader(new
FileReader("drinkLog.txt"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split("\t");
                if (parts.length < 2) continue;

                try {
                    LocalDateTime dateTime = LocalDateTime.parse(parts[0],
                            DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));

                    String date = dateTime.toLocalDate().toString();
                    String month = dateTime.getYear() + "-" + String.format("%02d",
dateTime.getMonthValue());

                    dailySales.put(date, dailySales.getOrDefault(date, 0) + 1);
                    monthlySales.put(month, monthlySales.getOrDefault(month, 0) + 1);
```

```
                } catch (DateTimeParseException e) {
                    // 날짜 형식 오류 무시
                }
            }
        }
    }

    // 음료별 매출 데이터 처리
    private void processBeverageData(Map<String, Map<String, Integer>>
dailyBeverageSales,
                                     Map<String, Map<String, Integer>>
monthlyBeverageSales) throws IOException {
        try (BufferedReader reader = new BufferedReader(new
FileReader("drinkLog.txt"))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split("\t");
                if (parts.length < 2) continue;

                try {
                    LocalDateTime dateTime = LocalDateTime.parse(parts[0],
                            DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
                    String beverage = parts[1];

                    String date = dateTime.toLocalDate().toString();
                    String month = dateTime.getYear() + "-" + String.format("%02d",
dateTime.getMonthValue());

                    // 일별 데이터 처리
                    dailyBeverageSales
                            .computeIfAbsent(date, k -> new HashMap<>())
                            .merge(beverage, 1, Integer::sum);

                    // 월별 데이터 처리
                    monthlyBeverageSales
                            .computeIfAbsent(month, k -> new HashMap<>())
                            .merge(beverage, 1, Integer::sum);

                } catch (DateTimeParseException e) {
                    // 날짜 형식 오류 무시
                }
            }
        }
    }


    // 파일에서 아이디와 비밀번호 로드
    private void loadCredentials() {
        try (BufferedReader reader = new BufferedReader(new
FileReader(CREDENTIALS_FILE))) {
            currentId = reader.readLine();
            currentPassword = reader.readLine();
        } catch (IOException e) {
            // 파일이 없으면 기본값 사용
            currentId = "admin";
            currentPassword = "Admin123!";
            saveCredentials();
        }
    }

    // 아이디와 비밀번호 파일에 저장
    private void saveCredentials() {
```

```
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(CREDENTIALS_FILE))) {
            writer.write(currentId + "\n" + currentPassword);
        } catch (IOException e) {
            JOptionPane.showMessageDialog(this,
                    "정보 저장 실패: " + e.getMessage(), "오류",
                    JOptionPane.ERROR_MESSAGE);
        }
    }
```

참고 문헌

[1] 송혜민, “nownews,” now news, 13 09 2013. [온라인]. Available: https://nownews.seoul.co.kr/news/newsView.php?id=20130913601006. [액세스: 6 4 2025].

[2] 로. C.마틴, 클린 아키텍처: 소프트웨어 구조와 설계의 원칙, 인사이트, 2019.

[3] itcode.dev, “itcode,” 15 8 2021. [온라인]. Available: https://blog.itcode.dev/posts/2021/08/15/liskov-subsitution-principle. [액세스: 7 4 2025].

[4] 불곰, “tistory,” tistory, 15 8 2021. [온라인]. Available: https://brownbears.tistory.com/577. [액세스: 10 4 2025].

[5] h. Kwon, “github,” 4 7 2018. [온라인]. Available: https://gmlwjd9405.github.io/2018/07/04/class-diagram.html. [액세스: 11 4 2025].