객체를 활용한 자동 판매기 프로그램 제작

학번: 2401110252

이름: 박지수

보고서 목차

1. 개요 2

가) 요구사항 2

나) 진행방식 2

다) 코드 설명 2

2. 코드 4

3. 결과 스크린샷 14

4. 고찰 21

1. 개요

가) 요구사항

가-ㄱ)

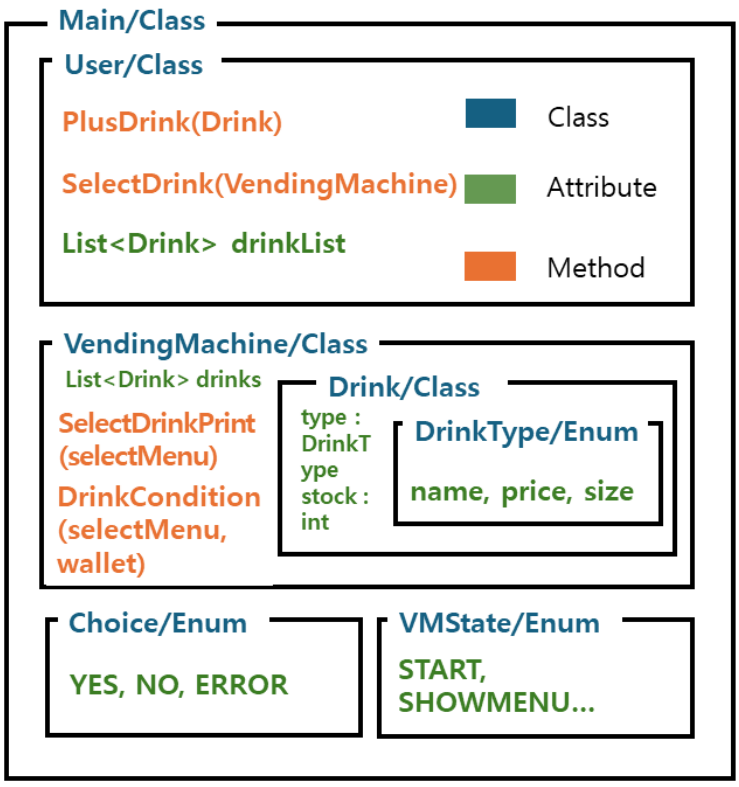
클래스를 사용하여 자판기와 음료수 객체를 생성하고 음료수를 구입한다.

나) 진행방식

나-ㄱ)

Main 클래스에서 User(사용자)객체가 키보드 입력을 통해 음료수를 구입하면 enum VMState 객체를 통해 자판기 구매 상태를 제어한다. 구입 조건은 재고의 여유가 있거나 사용자의 금액이 충분할 때이다. 구매 도중 장바구니를 통해 어떤 음료수를 샀는지 확인할 수 있으며 금액이 남아있더라도 구입을 중단할 수 있다. 구입을 끝마친 경우 구매하였던 음료수와 총 지불했던 가격을 표시하고 프로그램이 종료된다.

다) 코드 설명



주요 속성과 메서드를 표시하였다.

다-ㄱ) Drink/Class

음료수의 이름, 가격, 크기와 같은 제원은 enum class인 DrinkType을 통해 가져온 다음 계속 변경되는 재고 속성만 Drink 클래스에서 관리한다.

다-ㄴ) Drink/Enum

음료수의 이름, 가격, 크기가 저장 되어있다.

다-ㄷ) User/Class

음료수를 구매하기 위한 금액을 저장하는 wallet 속성과 구매한 음료수를 담아두기 위한 drinkList가 속성으로 포함 되어있다. 구매한 음료수를 장바구니에 넣는 PlusDrink 함수와 의사를 입력받는 SelectDrink ,SuggestShowDrinkList, SustainPurchase, ShowDrink, ShowResult 함수들이 있다.

다-ㄹ) VMState/Enum

Main 클래스에서 While문을 제어하기 위한 속성들이 있다.

다-ㅁ) VendingMachine/Class

Drinks 리스트를 속성에 넣어 판매하고 있는 음료수를 관리하고 DrinkCondition 함수를 통해서 구매가 가능한 환경인지 확인한다.

다-ㅂ) Choice/Enum

Main 클래스에서 사용자의 입력된 값을 반환하는 함수를 사용되어 그 값을 긍정과 부정, 에러를 표시 하기위한 Enum클래스이다.

다-ㅅ) Main/Class

While문과 Switch 문을 통해 계속 진행되며 사용자가 입력한 값에 따라 currentState 변수값을 결정한다. 프로그램 종료 전 단계는 Result 단계로 구매했던 물품과 동시에 currentState 값을 Done 단계로 변경하고 프로그램이 종료된다.

2. 코드

```
//24_06_19_알고리즘_실습과제
//한국폴리텍대학_서울정수캠퍼스_인공지능소프트웨어과
//2401110252 박지수
//객체를 활용한 자동판매기 제작

import java.util.List;
import java.util.ArrayList;
import java.util.Scanner;

public class User
{
    private int wallet;
    List<Drink> drinkList;
    Scanner sc;

    public User()
    {
        sc = new Scanner(System.in);
        wallet = 10000;
        drinkList = new ArrayList<>();
    }
    public int GetWallet()
    {
        return wallet;
    }
    public void ShowWallet()
    {
        System.out.printf("잔액: %d 원\n",GetWallet());
    }
    public void SetWallet(int price)
    {
        wallet -= price;
    }

    // 잘못된 입력을 하여 음료수에서 나온 drink 객체가 NULL 이 아닌 경우 함수가 진행.
    // 사용자 User 가 가지고 있는 drinkList 중에 자판기에서 건네받은 drinkType 과 지금
  // 소유하고있는 음료수 중의 drinkType 이 일치하다면 처음 사지 않았다는 first 지역변수
값을
  // false 로 변경하고 가지고 있는 drink 객체의 stock 속성을 1 올려주는 StockPlus
함수를 실행한다.
    // 만약 drinkList 가 비었고 first 값이 true 일때는 drinkList 에 새로 추가한다.
    public void PlusDrink(Drink drink)
    {
        if (drink == null)
        {
            return;
        }
        boolean first = true;
        for (Drink d : drinkList)
        {
```

```
                if (d.GetType() == drink.GetType())
                {
                    first = false;
                    d.StockPlus(1);
                }
            }
            if(drinkList.isEmpty() || first)
            {
                drink = new Drink(drink.GetType(), 1);
                drinkList.add(drink);
            }
        }
        public int SelectDrink(VendingMachine vm)
        {
            int selectMenu = 0;

            try
            {
                vm.AlertMenuSize();
                selectMenu = sc.nextInt();
            }
            catch(Exception e)
            {
                sc.nextLine();
            }
            if (selectMenu > vm.drinks.size()+1 || selectMenu < 1)
            {
                vm.ExactChoose();
                return -1;
                //이 값이 VendingMachine 의 SelectDrinkPrint 함수로 넘어간다.
                // -1 이면 drink 객체를 NULL 을 반환하기로 함
         }
            else
            {
                return selectMenu;
            }
        }
        public int SuggestShowDrinkList()
        {

            int selectChoice = 0;
                try
                {
                    System.out.println("1. 장바구니 보기\t2. 계속 구입\t3. 구입
    종료");
                    selectChoice = sc.nextInt();
                }
                catch(Exception e)
                {
                    sc.nextLine();
                }
            return selectChoice;
        }
        public int SustainPurchase()
        {
```

```
            int selectChoice = 0;
                try
                {
                    System.out.println("1. 계속 구입\t2. 구입 종료");
                    selectChoice = sc.nextInt();
                }
                catch(Exception e)
                {
                    sc.nextLine();
                }
            return selectChoice;
        }
        public void ShowDrink()
        {

            if (!drinkList.isEmpty())
            {
                for(Drink drink : drinkList)
                {
                    System.out.printf("%s %d 개\t", drink.GetName(),
    drink.GetStock());
                }
                System.out.println();
            }
            else
            {
                System.out.println("아무것도 사지않았습니다.");
            }
        }
        public void ShowResult()
        {
            int resultPrice =0;
            for(Drink drink : drinkList)
            {
                System.out.printf("%s %d 개 %d 원\t",
    drink.GetName(),drink.GetStock(), drink.GetStock()*drink.GetPrice());
                resultPrice+=drink.GetPrice()*drink.GetStock();
            }
            System.out.println();
            System.out.printf("총 %d 원", resultPrice);
        }
    }
```

User Class

```
public class Drink
{
    private final DrinkType type;
    private int stock;
    public Drink(DrinkType type, int stock)
    {
        this.type = type;
        this.stock = stock;
    }
```

```
    public int GetPrice()
    {
        return this.type.GetPrice();
    }
    public int GetStock()
    {
        return this.stock;
    }
    public DrinkType GetType()
    {
        return this.type;
    }
    public String GetName()
    {
        return this.type.GetName();
    }
    public int GetSize()
    {
        return this.type.GetSize();
    }
    public void SellDrink(int number)
    {
        stock -= number;
    }
    public void StockPlus(int number)
    {
        stock += number;
    }
    public void SetStock(int number)
    {
        stock = number;
    }
}
```

Drink Class

```
public enum DrinkType
{
    WATER( "Water",700, 500),
    COLA("Cola", 1000, 355 ),
    CIDER("Cider", 1500, 500 ),;

    private final String name;
    private final int price;
    private final int size;

    DrinkType(String name, int price, int size)
    {
        this.name = name;
        this.price = price;
        this.size = size;
    }

    public String GetName()
    {
        return name;
    }
    public int GetPrice()
    {
        return price;
```

```
    }
    public int GetSize()
    {
        return size;
    }
}
```

DrinkType Enum

```
import java.util.ArrayList;
import java.util.List;

public class VendingMachine
{
    List<Drink> drinks;
    public VendingMachine()
    {
        drinks = new ArrayList<>();
        drinks.add(new Drink(DrinkType.WATER,3));
        drinks.add(new Drink(DrinkType.COLA,3));
        drinks.add(new Drink(DrinkType.CIDER,3));
    }
    public int GetMinPrice()
    {
        int minPrice = drinks.get(0).GetPrice();
        for(Drink d : drinks)
        {
            if ( d.GetPrice() < minPrice)
            {
                minPrice = d.GetPrice();
            }
        }
        return minPrice;
    }
    public void ShowMenu()
    {
        for (int i = 0 ; i < drinks.size(); i++)
        {
            System.out.printf("%d.\t%s\t%dml\t%d 원\t 현재 재고 %d\n",i+1,
                    drinks.get(i).GetName(),drinks.get(i).GetSize(),
drinks.get(i).GetPrice(),drinks.get(i).GetStock());
        }
        System.out.println("4. \t 구입 종료.");
    }

    public int SelectDrinkPrint(int selectMenu)
    {
        if (selectMenu == -1)
        {
            return -1;
            //DrinkCondition 함수에 값이 넘어가고
       //-1 값을 받을 경우 Drink 객체를 NULL 반환한다.
        }
        DrinkType drinkType = drinks.get(selectMenu-1).GetType();

        switch (drinkType)
        {
```

```
            case WATER: // 1
                System.out.print("Water ");
                break;
            case COLA:
                System.out.print("Cola ");
                break;
            case CIDER:
                System.out.print("Cider ");
                break;
        }
        System.out.println("제품을 선택하였습니다.");
       return selectMenu-1;
    }
    public Drink DrinkCondition(int selectMenu, int wallet)
    {
        if (selectMenu == -1)
        {
            return null;
            // 이 NULL 값은 User 의 DrinkPlus 함수로 넘어간다.
        }
        Drink drink = drinks.get(selectMenu);

        if((wallet>=drink.GetPrice()) & drink.GetStock() > 0)
        {
            drink.SellDrink(1);
            System.out.printf("%s 를 구입하였습니다.\n",drink.GetName());
            return drink;
        }
        else if (wallet>=drink.GetPrice())
        {
            System.out.printf("%s 의 재고가 부족하다\n",drink.GetName());
            return null;
        }
        else if (drink.GetStock() > 0)
        {
            System.out.println("잔액이 부족하다.");
            return null;
        }
        else
        {
            System.out.printf("%s 을(를) 사기에는 재고와 잔액이
부족하다.\n",drink.GetName());
            return null;
        }
    }
    public void AlertMenuSize()
    {
        System.out.printf("1 ~ %d 까지 입력해 주세요\n",drinks.size()+1);
    }
    public void ExactChoose()
    {
        System.out.println("알맞은 입력을 해주세요");
    }
    public int GetDrinkListSize()
```

```
    {
        return drinks.size();
    }
    public boolean CheckStock()
    {
        for(Drink d : drinks)
        {
            if (d.GetStock()> 0)
            {
                return true;
            }
        }
        return false;
    }
}
```

VendingMachine Class

```
public class Main
{
    public static void main(String[] args)
    {
        VendingMachine vm = new VendingMachine();
        User user = new User();
        int userSelectMenuNum = 0;
        int inputVMNum = 0;
        int showDrinkList = 0;

        Drink chooseDrink;
        Choice choice;
        VMState currentState = VMState.START;
        while(currentState != VMState.DONE)
        {
            switch(currentState)
            {
                case START:
                    System.out.println("자판기 가동");
                case SHOWMENU:
                    if (!vm.CheckStock())
                    {
                        currentState = VMState.RESULT;
                        System.out.println("재고가 없어 구입이 종료됩니다.");
                        continue;
                    }
                    user.ShowWallet();
                    vm.ShowMenu();
                    currentState = VMState.SELECTMENUUSER;
                    break;
                case SELECTMENUUSER:
                    userSelectMenuNum = user.SelectDrink(vm);
                    if (vm.GetDrinkListSize()+1 == userSelectMenuNum)
                    {
                        currentState = VMState.RESULT;
                        continue;
                    }
                    currentState = VMState.INPUTNUMBERVMMACHINE;
                    break;
                case INPUTNUMBERVMMACHINE:
```

```
                    inputVMNum = vm.SelectDrinkPrint(userSelectMenuNum);
                    currentState = VMState.PURCHASE;
                    break;
                case PURCHASE:
                    chooseDrink = vm.DrinkCondition(inputVMNum,
user.GetWallet());
                    user.PlusDrink(chooseDrink);
                    if (chooseDrink == null)
                    {
                        if (user.GetWallet() < vm.GetMinPrice())
                        {
                            currentState = VMState.RESULT;
                            continue;
                        }
                        currentState = VMState.SELECTMENUUSER;
                        continue;
                    }
                    user.SetWallet(chooseDrink.GetPrice());
                    currentState = VMState.SELECTSHOWDRINKLIST;
                    break;
                case SELECTSHOWDRINKLIST:
                    // 숫자 이외의 입력은 user.SuggestShowDrinkList() 여기서 검사함.
                    showDrinkList = user.SuggestShowDrinkList();
                    choice = Choice.fromInt(showDrinkList,2);
                    switch(choice)
                    {
                        case YES_1:
                            currentState = VMState.SHOWDRINK;
                            break;
                        case YES_2:
                            currentState = VMState.SHOWMENU;
                            break;
                        case NO:
                            currentState = VMState.RESULT;
                            break;
                        case ERROR:
                            vm.ExactChoose();
                            break;
                    }
                    break;
                case SHOWDRINK:
                    user.ShowDrink();
                    currentState = VMState.PURCHASEDONE;
                break;
                case PURCHASEDONE:
                    // 숫자 이외의 입력은 user.SustainPurchase() 여기서 검사함.
                    choice = Choice.fromInt(user.SustainPurchase(),1);
                    switch(choice)
                    {
                        case YES_1:
                            currentState = VMState.SHOWMENU;
                            break;
                        case NO:
                            currentState = VMState.RESULT;
                            break;
                        case ERROR:
                            vm.ExactChoose();
```

```
                            break;
                        }
                        break;
                    case RESULT:
                        System.out.println("구입 종료");
                        user.ShowResult();
                        currentState = VMState.DONE;
                        break;
                }
            }
        }
}
```

Main Class

```
public enum Choice {
    YES_1(1),
    YES_2(2),
    YES_3(3),
    NO(4),
    ERROR(-1);

    private final int choiceNum;

    Choice(int choiceNum) {
        this.choiceNum = choiceNum;
    }

    public int getChoiceNum() {
        return choiceNum;
    }

    // 번호의 마지막은 탈출하려는 부정의 선택지이며 그 전까지는 긍정의 다른 선택지이다.
    // 긍정의 선택지 가짓수를 limitYesNum 파라미터에 넣고 사용자가 입력받은 숫자를
choiceNum 에 넣는다.
    // 긍정의 선택 가짓수 보다 1 높은 숫자는 부정의 선택 탈출을 의미하여 첫 if 문에서 검사 후
반복문을 진행한다.
    // 첫 조건문, 반복문에도 걸리지않는다면 그 외의 숫자를 입력한 것으로 ERROR 를 반환하게
된다.
    public static Choice fromInt(int choiceNum,int limitYesNum)
    {
        int count = 0;
        if(choiceNum == limitYesNum+1)
        {
            return NO;
        }
        for (Choice choice : Choice.values())
        {
            if (count == limitYesNum)
            {
                break;
            }
            if (choice.getChoiceNum() == choiceNum) {
                return choice;
            }
```

```
            count++;
        }
        return ERROR;
    }
}
```

Choice Enum

```
public enum VMState
{
    START,
    SHOWMENU,
    SELECTMENUUSER,
    INPUTNUMBERVMMACHINE,
    PURCHASE,
    SHOWDRINK,
    SELECTSHOWDRINKLIST,
    PURCHASEDONE,
    RESULT,
    DONE
}
```

VMState Enum

3. 결과 스크린샷

```
자판기 가동
잔액: 10000원
1.  Water   500ml   700원   현재 재고 3
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 9300원
1.  Water   500ml   700원   현재 재고 2
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 8600원
1.  Water   500ml   700원   현재 재고 1
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 7900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
2
Cola 제품을 선택하였습니다.
Cola를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 6900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 2
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
2
Cola 제품을 선택하였습니다.
Cola를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 5900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 1
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
2
Cola 제품을 선택하였습니다.
Cola를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 4900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 0
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
```

```
1 ~ 4 까지 입력해 주세요
3
Cider 제품을 선택하였습니다.
Cider를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 3400원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 0
3.  Cider   500ml   1500원  현재 재고 2
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
3
Cider 제품을 선택하였습니다.
Cider를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 1900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 0
3.  Cider   500ml   1500원  현재 재고 1
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
3
Cider 제품을 선택하였습니다.
Cider를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
재고가 없어 구입이 종료됩니다.
구입 종료
Water 3개 2100원 Cola 3개 3000원  Cider 3개 4500원
총 9600원
```

모든 음료수를 다 샀을 경우

```
자판기 가동
잔액: 10000원
1.  Water   500ml   700원   현재 재고 3
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
1
Water 1개
1. 계속 구입  2. 구입 종료
1
잔액: 9300원
1.  Water   500ml   700원   현재 재고 2
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
2
Cola 제품을 선택하였습니다.
Cola를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
1
Water 1개   Cola 1개
1. 계속 구입  2. 구입 종료
1
잔액: 8300원
1.  Water   500ml   700원   현재 재고 2
2.  Cola    355ml   1000원  현재 재고 2
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
3
Cider 제품을 선택하였습니다.
Cider를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
1
Water 1개   Cola 1개    Cider 1개
1. 계속 구입  2. 구입 종료
1
잔액: 6800원
1.  Water   500ml   700원   현재 재고 2
2.  Cola    355ml   1000원  현재 재고 2
3.  Cider   500ml   1500원  현재 재고 2
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
1
Water 2개   Cola 1개    Cider 1개
1. 계속 구입  2. 구입 종료
```

구입 도중 장바구니를 볼 경우

```
자판기 가동
잔액: 10000원
1.  Water   500ml   700원   현재 재고 3
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 9300원
1.  Water   500ml   700원   현재 재고 2
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
4
구입 종료
Water 1개 700원
총 700원
```

```
자판기 가동
잔액: 10000원
1.  Water   500ml   700원   현재 재고 3
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
3
구입 종료
Water 1개 700원
총 700원
```

구입 도중 구입 종료

```
자판기 가동
잔액: 10000원
1.  Water   500ml   700원   현재 재고 3
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 9300원
1.  Water   500ml   700원   현재 재고 2
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 8600원
1.  Water   500ml   700원   현재 재고 1
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 7900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water의 재고가 부족하다
1 ~ 4 까지 입력해 주세요
2
Cola 제품을 선택하였습니다.
Cola를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 6900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 2
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
2
Cola 제품을 선택하였습니다.
Cola를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 5900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 1
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
2
Cola 제품을 선택하였습니다.
Cola를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 4900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 0
3.  Cider   500ml   1500원  현재 재고 3
```

```
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
2
Cola 제품을 선택하였습니다.
Cola의 재고가 부족하다
1 ~ 4 까지 입력해 주세요
3
Cider 제품을 선택하였습니다.
Cider를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 3400원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 0
3.  Cider   500ml   1500원  현재 재고 2
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
3
Cider 제품을 선택하였습니다.
Cider를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 1900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 0
3.  Cider   500ml   1500원  현재 재고 1
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
3
Cider 제품을 선택하였습니다.
Cider를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
재고가 없어 구입이 종료됩니다.
구입 종료
Water 3개 2100원 Cola 3개 3000원  Cider 3개 4500원
총 9600원3
```

재고 부족의 경우

```
자판기 가동
잔액: 10000원
1.  Water   500ml   700원   현재 재고 3
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
5
알맞은 입력을 해주세요
1 ~ 4 까지 입력해 주세요
▱
알맞은 입력을 해주세요
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
4
알맞은 입력을 해주세요
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
▱
알맞은 입력을 해주세요
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
1
Water 1개
1. 계속 구입  2. 구입 종료
3
알맞은 입력을 해주세요
1. 계속 구입  2. 구입 종료
▱
알맞은 입력을 해주세요
1. 계속 구입  2. 구입 종료
1
잔액: 9300원
1.  Water   500ml   700원   현재 재고 2
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 8600원
1.  Water   500ml   700원   현재 재고 1
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
2
잔액: 7900원
1.  Water   500ml   700원   현재 재고 0
2.  Cola    355ml   1000원  현재 재고 3
3.  Cider   500ml   1500원  현재 재고 3
4.  구입 종료.
1 ~ 4 까지 입력해 주세요
1
Water 제품을 선택하였습니다.
Water의 재고가 부족하다
1 ~ 4 까지 입력해 주세요
5
알맞은 입력을 해주세요
1 ~ 4 까지 입력해 주세요
▱
알맞은 입력을 해주세요
1 ~ 4 까지 입력해 주세요
2
Cola 제품을 선택하였습니다.
Cola를 구입하였습니다.
1. 장바구니 보기  2. 계속 구입  3. 구입 종료
1
```

```
Water 3개    Cola 1개
1. 계속 구입  2. 구입 종료
3
알맞은 입력을 해주세요
1. 계속 구입  2. 구입 종료
▱
알맞은 입력을 해주세요
1. 계속 구입  2. 구입 종료
2
구입 종료
Water 3개 2100원 Cola 1개 1000원
총 3100원
```

오입력의 경우

4. 고찰

지난 마법사 과제에서 상태를 제어하기 위해서 while문과 Switch문을 이용해 작성하였지만, 점점 단계가 많아지고 복잡해지니 이때 쉽게 이해하려고 표시한 단계표시가 더 읽어내기 어려웠다. 그래서 6월 11일에 배운 Enum 클래스를 이용하여 상태를 문자로 명시해 주면 더욱 가독성이 좋아질 것 같아 사용하였다. 하지만 사용방법이 익숙하지 않아 완벽히 활용하지는 못했지만, DrinkType에서 사용할때에는 적절히 쓴 것 같다. 그 예로 음료수의 이름과 가격은 변경되지 않으니, Java의 Enum을 이용해 추가 정보를 넣어 재고만 Drink 클래스에서 관리하게 설정하였다.

마법사 과제에서도 괜히 재귀함수로 설정하여 고생했기 때문에 함수 안에서 다시 진입하지 않고 Main의 진행 순서를 다시 반복하는 형식으로 변경하니 복잡하게 생각하지 않아도 되어 스택 프레임의 무덤에서 벗어날 수 있었다.

마지막으로 처음 사용자가 구매한 음료수를 담기 위해서 자판기의 음료 그 자체를 추가한 실수가 있었다. 당연히 그 객체를 가져오니 재고가 공유되어 한 개를 구매하였을 때 자판기의 재고 역시 한 개로 변경 되어있었다. 수정하기 위해서 새로 객체를 생성한 다음 리스트에 넣어주었다. 똑같은 음료를 살 경우에는 Drink 객체를 검사하지 않고 Drink객체의 DrinkType을 검사하여 이미 중복된 음료수일 경우 재고만 추가하는 방식으로 설정하였다. 그 결과 같은 것을 여러 개 구매해도 새로 만든 Drink 객체의 재고만 올릴 수 있었다.

과제를 계속 진행하면서 더 많은 코드를 작성하고 조금씩 새로운 것을 습득하려다 보니 성장하지 않은 것 같으면서도 조금씩 진전하는 것 같다.