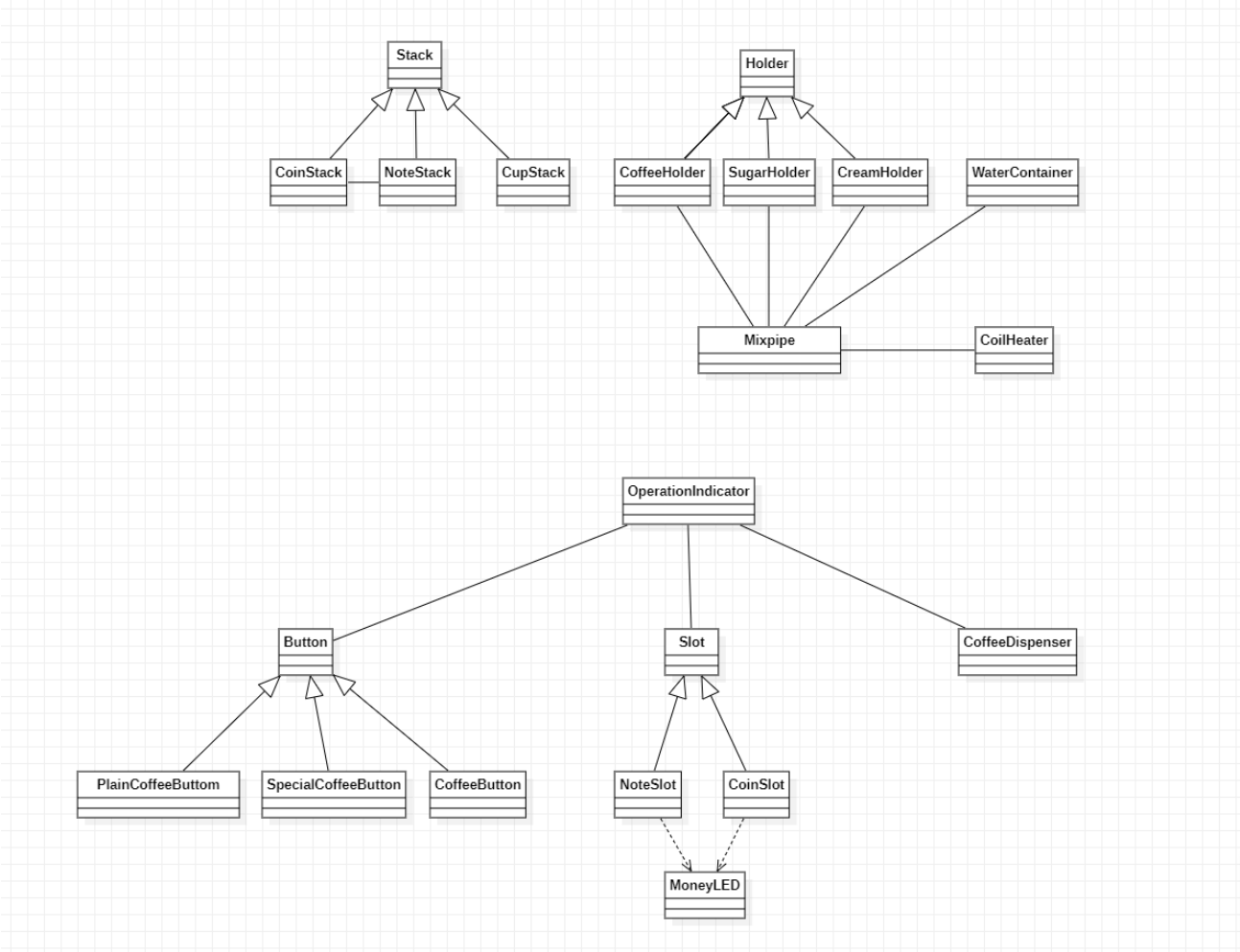
소프트웨어 분석 및 설계

자판기 클래스 모델 구현

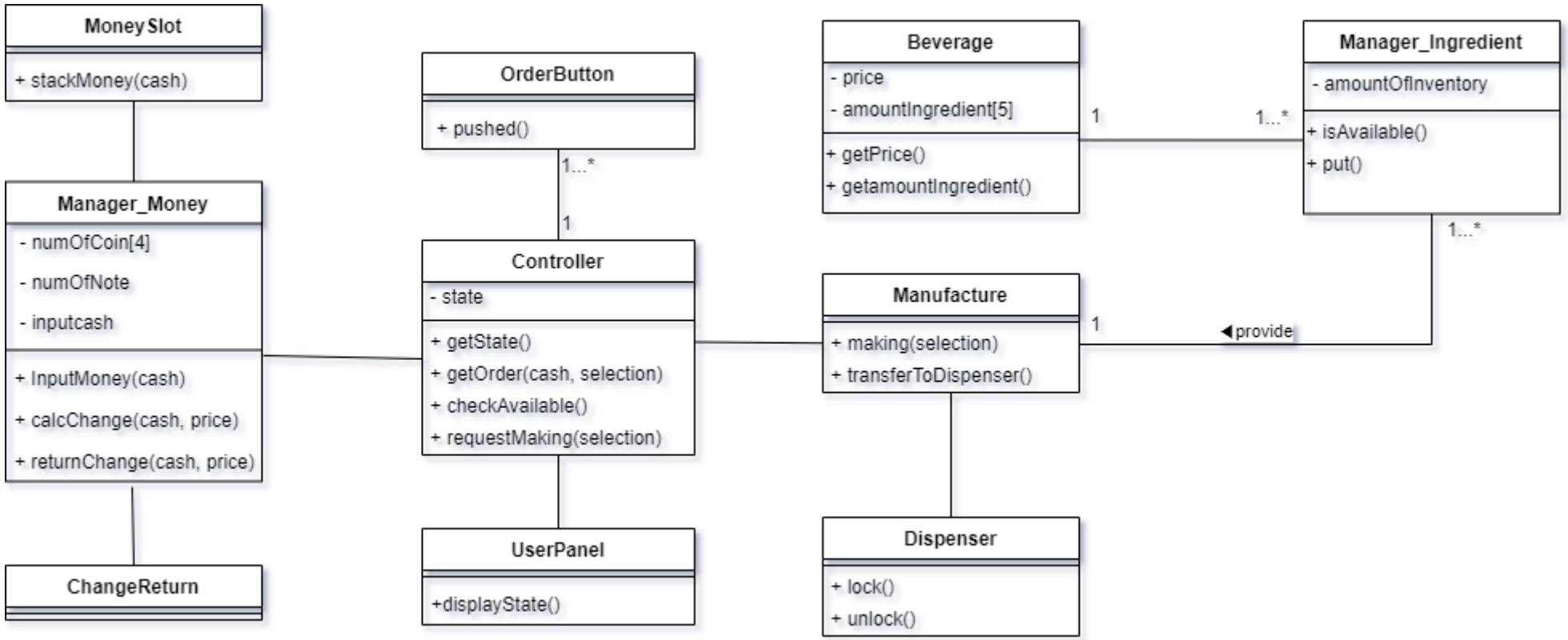
사용언어: C++

20190415 황혜령

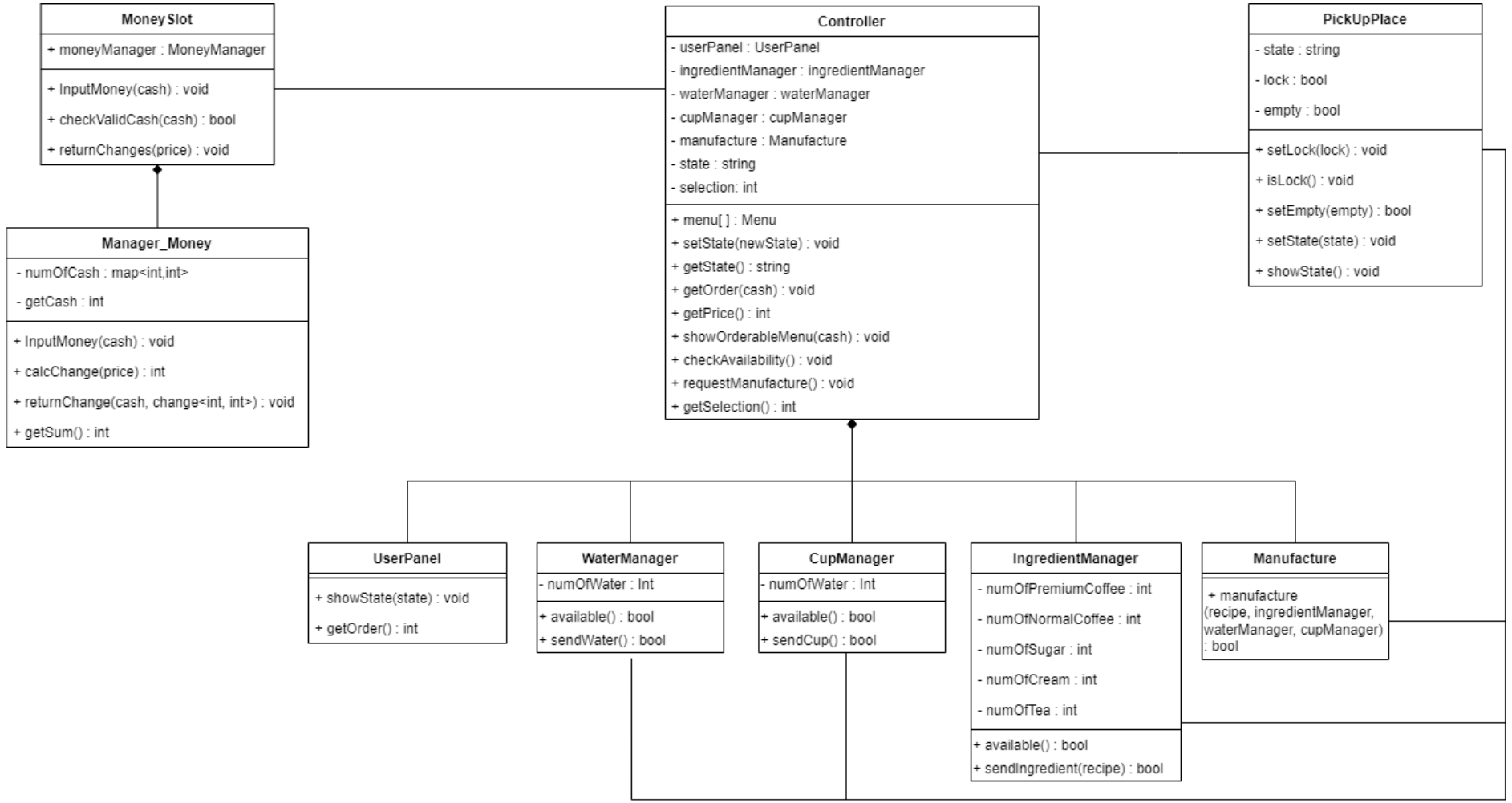
1. 클래스 모델 _수정단계(4주차)



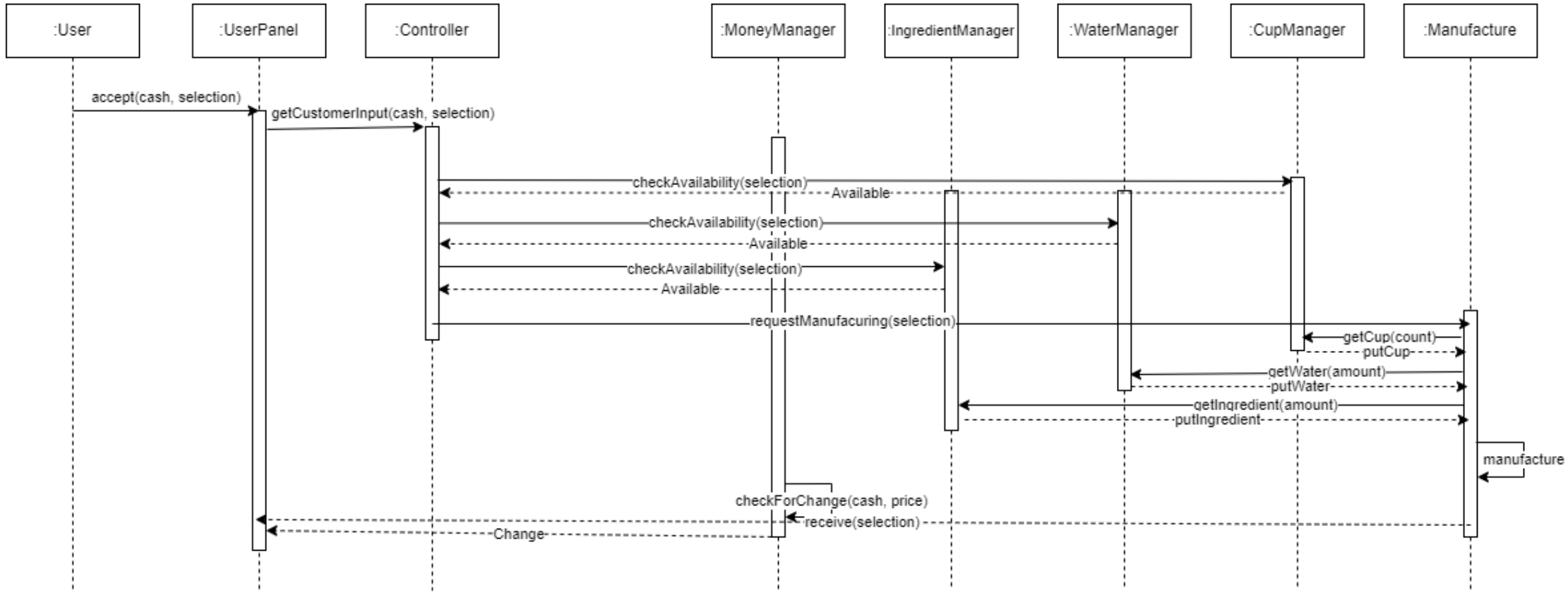
1. 클래스 모델 _수정단계(10주차)



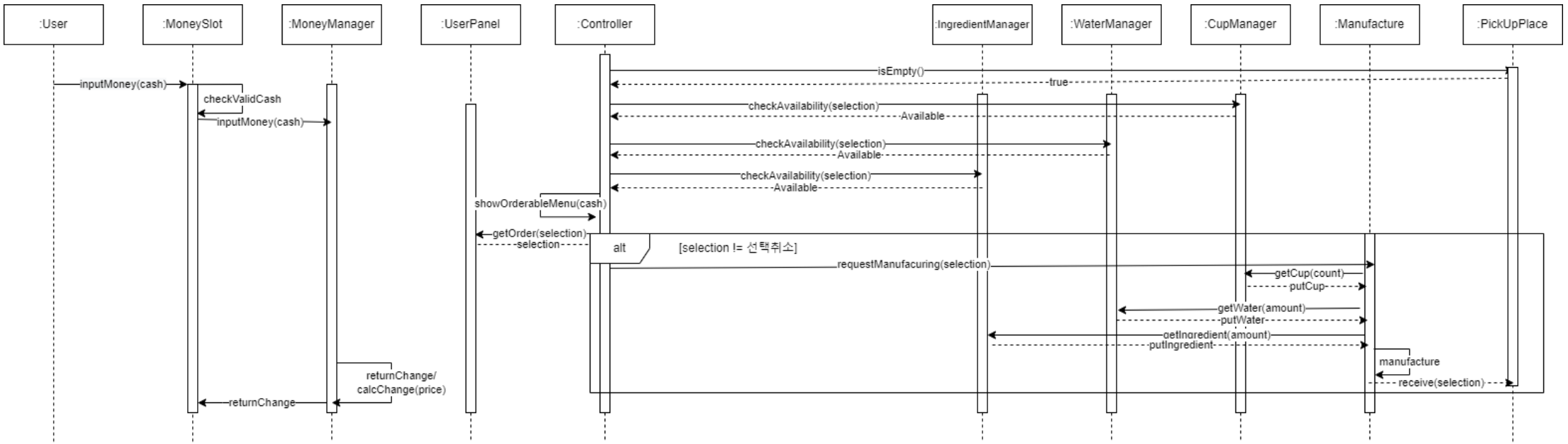
1. 클래스 모델 _최종 버전


MoneySlot
+ moneyManager : MoneyManager
+ InputMoney(cash) : void
+ checkValidCash(cash) : bool
+ returnChanges(price) : void
Manager_Money
- numOfCash : map<int,int>
- getCash : int
+ InputMoney(cash) : void
+ calcChange(price) : int
+ returnChange(cash, change<int, int>) : void
+ getSum() : int
Controller
- userPanel : UserPanel
- ingredientManager : ingredientManager
- waterManager : waterManager
- cupManager : cupManager
- manufacture : Manufacture
- state : string
- selection: int
+ menu[ ] : Menu
+ setState(newState) : void
+ getState() : string
+ getOrder(cash) : void
+ getPrice() : int
+ showOrderableMenu(cash) : void
+ checkAvailability() : void
+ requestManufacture() : void
+ getSelection() : int
PickUpPlace
- state : string
- lock : bool
- empty : bool
+ setLock(lock) : void
+ isLock() : void
+ setEmpty(empty) : bool
+ setState(state) : void
+ showState() : void
UserPanel
+ showState(state) : void
+ getOrder() : int
WaterManager
- numOfWater : Int
+ available() : bool
+ sendWater() : bool
CupManager
- numOfWater : Int
+ available() : bool
+ sendCup() : bool
IngredientManager
- numOfPremiumCoffee : int
- numOfNormalCoffee : int
- numOfSugar : int
- numOfCream : int
- numOfTea : int
+ available() : bool
+ sendIngredient(recipe) : bool
Manufacture
+ manufacture (recipe, ingredientManager, waterManager, cupManager) : bool

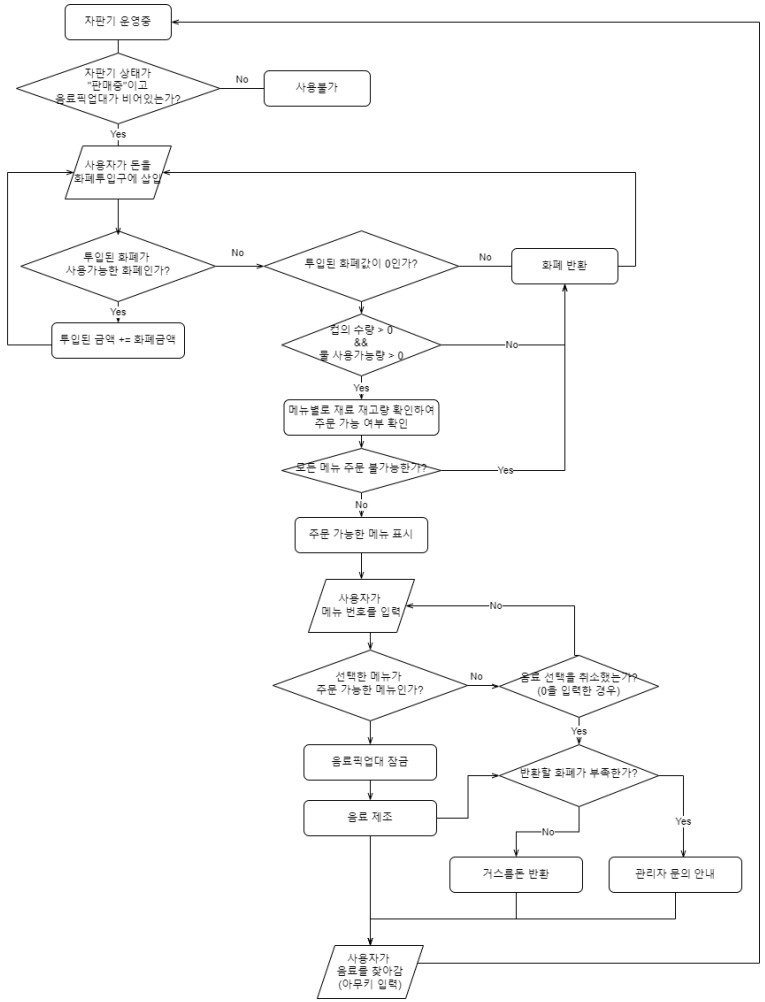
2. 시퀀스 모델_수정단계(9주차)



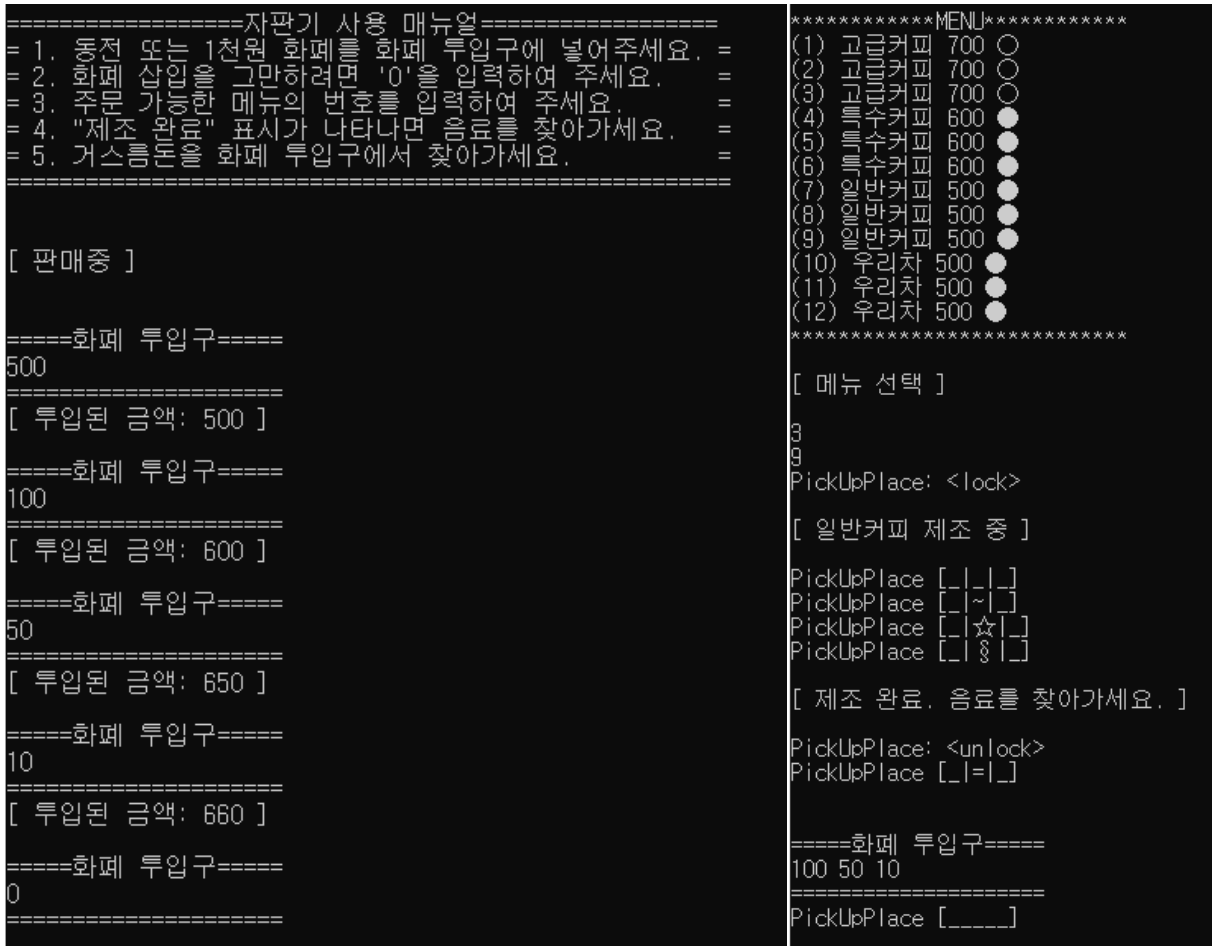
2. 시퀀스 모델 _최종 버전


:User
:MoneySlot
:MoneyManager
:UserPanel
:Controller
:IngredientManager
:WaterManager
:CupManager
:Manufacture
:PickUpPlace
isEmpty()
true
inputMoney(cash)
checkValidCash
inputMoney(cash)
checkAvailability(selection)
Available
checkAvailability(selection)
Available
checkAvailability(selection)
Available
showOrderableMenu(cash)
getOrder(selection)
selection
alt
[selection != 선택취소]
requestManufacuring(selection)
getCup(count)
putCup
getWater(amount)
putWater
getIngredient(amount)
putIngredient
manufacture
receive(selection)
returnChange/
calcChange(price)
returnChange

3. 구현 코드 흐름도

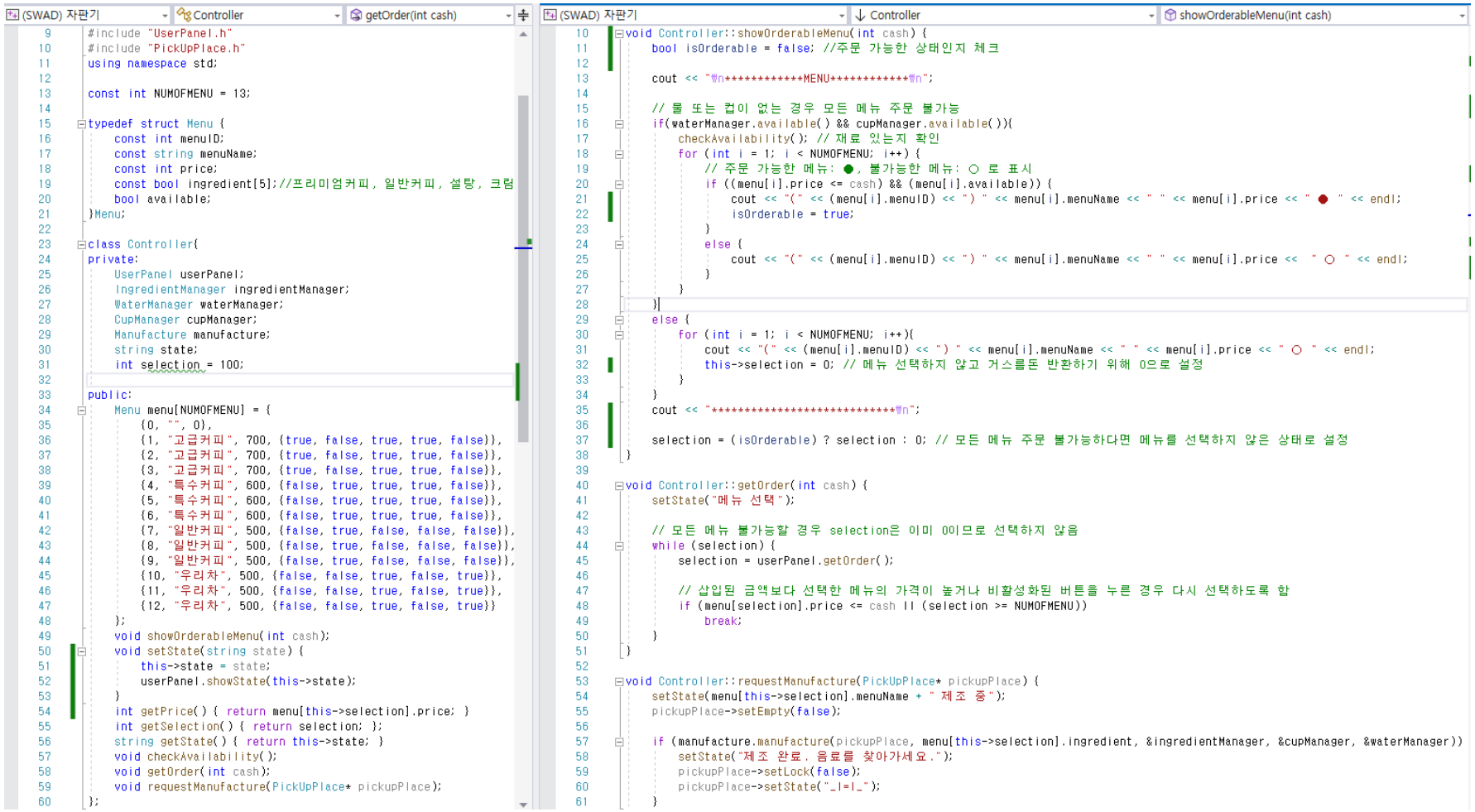

자판기 운영중
자판기 상태가 "판매중"이고 음료픽업대가 비어있는가?
No
사용불가
Yes
사용자가 돈을 화폐투입구에 삽입
투입된 화폐가 사용가능한 화폐인가?
No
투입된 화폐값이 0인가?
No
화폐 반환
Yes
투입된 금액 += 화폐금액
컵의 수량 > 0 && 물 사용가능량 > 0
No
Yes
메뉴별로 재료 재고량 확인하여 주문 가능 여부 확인
모든 메뉴 주문 불가능한가?
Yes
No
주문 가능한 메뉴 표시
사용자가 메뉴 번호를 입력
No
선택한 메뉴가 주문 가능한 메뉴인가?
No
음료 선택을 취소했는가? (0을 입력한 경우)
Yes
음료픽업대 잠금
반환할 화폐가 부족한가?
음료 제조
No
Yes
거스름돈 반환
관리자 문의 안내
사용자가 음료를 찾아감 (아무키 입력)

4. 실행화면



정상적으로 buy coffee 유스케이스가 작동되는 경우의 실행 화면입니다.

5. 소스코드_Controller Class



```
(SWAD) 자판기 | Controller | getOrder(int cash)

#include "UserPanel.h"
#include "PickUpPlace.h"
using namespace std;

const int NUMOFMENU = 13;

typedef struct Menu {
    const int menuID;
    const string menuName;
    const int price;
    const bool ingredient[5];//프리미엄커피, 일반커피, 설탕, 크림
    bool available;
}Menu;

class Controller{
private:
    UserPanel userPanel;
    IngredientManager ingredientManager;
    WaterManager waterManager;
    CupManager cupManager;
    Manufacture manufacture;
    string state;
    int selection = 100;

public:
    Menu menu[NUMOFMENU] = {
        {0, "", 0},
        {1, "고급커피", 700, {true, false, true, true, false}},
        {2, "고급커피", 700, {true, false, true, true, false}},
        {3, "고급커피", 700, {true, false, true, true, false}},
        {4, "특수커피", 600, {false, true, true, true, false}},
        {5, "특수커피", 600, {false, true, true, true, false}},
        {6, "특수커피", 600, {false, true, true, true, false}},
        {7, "일반커피", 500, {false, true, false, false, false}},
        {8, "일반커피", 500, {false, true, false, false, false}},
        {9, "일반커피", 500, {false, true, false, false, false}},
        {10, "우리차", 500, {false, false, true, false, true}},
        {11, "우리차", 500, {false, false, true, false, true}},
        {12, "우리차", 500, {false, false, true, false, true}}
    };
    void showOrderableMenu(int cash);
    void setState(string state) {
        this->state = state;
        userPanel.showState(this->state);
    }
    int getPrice() { return menu[this->selection].price; }
    int getSelection() { return selection; };
    string getState() { return this->state; }
    void checkAvailability();
    void getOrder(int cash);
    void requestManufacture(PickUpPlace* pickupPlace);
};
```

```
(SWAD) 자판기 | Controller | showOrderableMenu(int cash)

void Controller::showOrderableMenu(int cash) {
    bool isOrderable = false; //주문 가능한 상태인지 체크

    cout << "₩n++++++++++++MENU++++++++++₩n";

    // 물 또는 컵이 없는 경우 모든 메뉴 주문 불가능
    if(waterManager.available() && cupManager.available()){
        checkAvailability(); // 재료 있는지 확인
        for (int i = 1; i < NUMOFMENU; i++) {
            // 주문 가능한 메뉴: ●, 불가능한 메뉴: ○ 로 표시
            if ((menu[i].price <= cash) && (menu[i].available)) {
                cout << "(" << (menu[i].menuID) << ") " << menu[i].menuName << " " << menu[i].price << " ● " << endl;
                isOrderable = true;
            }
            else {
                cout << "(" << (menu[i].menuID) << ") " << menu[i].menuName << " " << menu[i].price <<  " ○ " << endl;
            }
        }
    }
    else {
        for (int i = 1; i < NUMOFMENU; i++){
            cout << "(" << (menu[i].menuID) << ") " << menu[i].menuName << " " << menu[i].price << " ○ " << endl;
            this->selection = 0; // 메뉴 선택하지 않고 거스름돈 반환하기 위해 0으로 설정
        }
    }
    cout << "**************************₩n";

    selection = (isOrderable) ? selection : 0; // 모든 메뉴 주문 불가능하다면 메뉴를 선택하지 않은 상태로 설정
}

void Controller::getOrder(int cash) {
    setState("메뉴 선택");

    // 모든 메뉴 불가능할 경우 selection은 이미 00이므로 선택하지 않음
    while (selection) {
        selection = userPanel.getOrder();

        // 삽입된 금액보다 선택한 메뉴의 가격이 높거나 비활성화된 버튼을 누른 경우 다시 선택하도록 함
        if (menu[selection].price <= cash || (selection >= NUMOFMENU))
            break;
    }
}

void Controller::requestManufacture(PickUpPlace* pickupPlace) {
    setState(menu[this->selection].menuName + " 제조 중");
    pickupPlace->setEmpty(false);

    if (manufacture.manufacture(pickupPlace, menu[this->selection].ingredient, &ingredientManager, &cupManager, &waterManager))
        setState("제조 완료. 음료를 찾아가세요.");
        pickupPlace->setLock(false);
        pickupPlace->setState("_I=I_");
    }
```

5. 소스코드_MoneySlot Class

(SWAD) 자판기 (전역 범위)

```
#ifndef MONEYSLOT_H
#define MONEYSLOT_H
#include <string>
#include "MoneyManager.h"

class MoneySlot{
public:
    MoneyManager moneyManager;
    bool checkVaildCash(int cash); // 사용가능한 화폐인지 검사
    void inputMoney(); // 사용자가 삽입한 화폐
    void returnChanges(int price); // 거스름돈 반환
};

#endif
```

(SWAD) 자판기 MoneySlot inputMoney()

```
bool MoneySlot::checkVaildCash(int cash) {
    if (cash == 10 || cash == 50 || cash == 100 || cash == 500 || cash == 1000)
        return true;
    else
        return false;
}

void MoneySlot::inputMoney() {
    int cash = 0;

    //사용자가 자판기에 화폐 투입
    while (true) {
        cout << "₩n=====화폐 투입구=====₩n"; cin >> cash; cout << "==================== " << endl;

        // 사용가능한 화폐이면 moneyManager가 관리하도록 함
        if (checkVaildCash(cash)) {
            moneyManager.inputMoney(cash);
        }
        // 사용 불가능한 화폐이면 반환
        else {
            if (cash == 5000 || cash == 10000 || cash == 50000){
                cout << "반환: " << cash << endl;
                cash = 0;
            }
            // 0을 입력했을시 화폐 삽입 멈춤
            else if (cash == 0)
                return;
        }
        cout << "[ 투입된 금액: " << moneyManager.getSum() << " ]" << endl;
    }
}

void MoneySlot::returnChanges(int price) {
    //화폐별 거스름돈 개수 저장 공간
    map<int, int> change = {
        {10, 0}, {50, 0}, {100, 0}, {500, 0}, {1000, 0}
    };
    moneyManager.returnChange(price, change);

    //moneySlot에 거스름돈 반환
    cout << "₩n=====화폐 투입구=====₩n";
    while (change[1000]){
        cout << 1000 << " ";
        change[1000]--;
    }
    while (change[500]) {
        cout << 500 << " ";
        change[500]--;
    }
    while (change[100]) {
        cout << 100 << " ";
        change[100]--;
```

5. 소스코드_MoneyManager Class

(SWAD) 자판기 | (전역 범위)

```
#ifndef MONEYMANAGER_H
#define MONEYMANAGER_H
#include <string>
#include <map>
using namespace std;

class MoneyManager{
private:
    map<int, int> numOfCash = {
        {10, 500}, {50, 500}, {100, 500}, {500, 500}, {1000, 500}
    };
    int getCash = 0;
public:
    void inputMoney(int cash);
    int calcChange(int price);
    void returnChange(int price, map<int, int>& change);
    int getSum() { return this->getCash; }
};

#endif
```

(SWAD) 자판기 | MoneyManager | returnChange(int price, map<int,int>& change)

```
void MoneyManager::inputMoney(int cash) {
    // 동전 또는 지폐 하나씩 넣으면서 금액 증가
    this->numOfCash[cash]++;
    this->getCash += cash;
}

int MoneyManager::calcChange(int price) {
    return (this->getCash - price);
}

void MoneyManager::returnChange(int price, map<int, int>& change) {
    int changes = this->calcChange(price);
    this->getCash -= price;

    while (changes > 0) {
        if ((changes >= 1000) && (numOfCash[1000] > 0)) {
            changes -= 1000;
            this->getCash -= 1000;
            this->numOfCash[1000]--;
            change[1000]++;
        }
        else if((changes >= 500) && (numOfCash[500] > 0)){
            changes -= 500;
            this->getCash -= 500;
            this->numOfCash[500]--;
            change[500]++;
        }
        else if ((changes >= 100) && (numOfCash[100] > 0)) {
            changes -= 100;
            this->getCash -= 100;
            this->numOfCash[100]--;
            change[100]++;
        }
        else if ((changes >= 50) && (numOfCash[50] > 0)) {
            changes -= 50;
            this->getCash -= 50;
            this->numOfCash[50]--;
            change[50]++;
        }
        else if(numOfCash[10] > 0) {
            changes -= 10;
            this->getCash -= 10;
            this->numOfCash[10]--;
            change[10]++;
        }
        else {
            std::cout << "₩n자판기 내 반환할 화폐가 부족합니다. 관리자에게 문의해주세요. 010-xxxx-xxxx" << endl;
            break;
        }
    }
    std::cout << endl;
}
```

5. 소스코드_IngredientManager Class

(SWAD) 자판기 (전역 범위)

```
1  #ifndef INGREDIENTMANAGER_H
2  #define INGREDIENTMANAGER_H
3  #include <string>
4  #include "PickUpPlace.h"
5  using namespace std;
6
7  class IngredientManager{
8  private:
9      int numOfPremiumCoffee = 565;
10     int numOfNormalCoffee = 600;
11     int numOfSugar = 618;
12     int numOfCream = 350;
13     int numOfTea = 175;
14
15 public:
16     bool available(const bool ingredient[5]);
17     bool sendIngredient(const bool ingredient[5], PickUpPlace* pi
18 };
19
20 #endif
```

(SWAD) 자판기 IngredientManager available(const bool ingredient[5])

```
6  bool IngredientManager::available(const bool ingredient[5]) {
7      bool isAvailable = false;
8
9      if (ingredient[0]){
10         if (numOfPremiumCoffee > 0)
11             isAvailable = true;
12         else
13             return false;
14     }
15     if (ingredient[1]) {
16         if (numOfNormalCoffee > 0)
17             isAvailable = true;
18         else
19             return false;
20     }
21     if (ingredient[2]) {
22         if (numOfSugar > 0)
23             isAvailable = true;
24         else
25             return false;
26     }
27     if (ingredient[3]) {
28         if (numOfCream > 0)
29             isAvailable = true;
30         else
31             return false;
32     }
33     if (ingredient[4]) {
34         if (numOfTea > 0)
35             isAvailable = true;
36         else
37             return false;
38     }
39     return isAvailable;
40 }
41 bool IngredientManager::sendIngredient(const bool ingredient[5], PickUpPlace* pickupPlace) {
42     if (ingredient[0]){
43         numOfPremiumCoffee--;
44         Sleep(1000);
45         pickupPlace->setState("_I★I_");
46     }
47     if (ingredient[1]){
```

5. 소스코드_CupManager Class

(SWAD) 자판기 | (전역 범위)

```
1  #ifndef CUPMANAGER_H
2  #define CUPMANAGER_H
3  #include <string>
4  #include "PickUpPlace.h"
5  using namespace std;
6
7  class CupManager{
8  private:
9      int numOfCup = 350;
10 public:
11     bool available();
12     bool sendCup(PickUpPlace*);
13 };
14
15 #endif
```

(SWAD) 자판기 | CupManager | sendCup(PickUpPlace * pickupPlace)

```
1  #include "CupManager.h"
2  #include <iostream>
3  #include <Windows.h>
4  using namespace std;
5
6  bool CupManager::available() {
7      return numOfCup;
8  }
9
10 bool CupManager::sendCup(PickUpPlace* pickupPlace) {
11     this->numOfCup--;
12     Sleep(1000);
13     pickupPlace->setState("_|_|_");
14     return true;
15 }
```

5. 소스코드_WaterManager Class

(SWAD) 자판기 (전역 범위)

```
#ifndef WATERMANAGER_H
#define WATERMANAGER_H
#include <string>
#include "PickUpPlace.h"
using namespace std;

class WaterManager{
private:
    int numOfWater = 340;
public:
    bool available();
    bool sendWater(PickUpPlace* pickupPlace);
};

#endif
```

(SWAD) 자판기 WaterManager sendWater(PickUpPlace * pickupPlace)

```
#include "WaterManager.h"
#include <iostream>
#include <Windows.h>
 using namespace std;

bool WaterManager::available() {
    return this->numOfWater;
}

bool WaterManager::sendWater(PickUpPlace* pickupPlace) {
    this->numOfWater--;
    Sleep(2000);
    pickupPlace->setState("_|~|_");
    return true;
}
```

5. 소스코드_UserPanel Class

(SWAD) 자판기 | UserPanel

```
1   #ifndef USERPANEL_H
2   #define USERPANEL_H
3
4   #include <iostream>
5   #include <string>
6   using namespace std;
7
8   class UserPanel{
9   public:
10      void showState(string state) { cout << "₩n[ " << state << " ]₩n₩n"; };
11      int getOrder() {
12          int select;
13          cin >> select;
14
15          return select;
16      }
17  };
18
19  #endif
```

5. 소스코드_Manufacture Class

(SWAD) 자판기 | Manufacture

```
1  #ifndef MANUFACTURE_H
2  #define MANUFACTURE_H
3  #include <iostream>
4  #include <string>
5  #include <Windows.h>
6  #include "IngredientManager.h"
7  #include "CupManager.h"
8  #include "WaterManager.h"
9  #include "Manufacture.h"
10 #include "pickUpPlace.h"
11 using namespace std;
12
13 class Manufacture{
14 public:
15     bool manufacture(PickUpPlace*, const bool*, IngredientManager*, CupManager*, WaterManager*);
16 };
17
18 bool Manufacture::manufacture(PickUpPlace* pickupPlace, const bool* recipe, IngredientManager* ingrediendtManager, CupManager* cupManager, WaterManager* waterManager) {
19     if (cupManager->sendCup(pickupPlace) && waterManager->sendWater(pickupPlace) && ingrediendtManager->sendIngredient(recipe, pickupPlace)) {
20         pickupPlace->setState("_ㅣ§ㅣ_"); //mix
21         Sleep(2000);
22         return true;
23     }
24     else false;
25 }
26
27 #endif
```

5. 소스코드_PickUpPlace Class

(SWAD) 자판기 | PickUpPlace

```
#ifndef PICKUPPLACE_H
#define PICKUPPLACE_H
#include <iostream>
#include <string>
using namespace std;

class PickUpPlace{
private:
    string state;
    bool lock = false;
    bool empty = true; // 픽업대에 음료 존재 여부
public:
    void setLock(bool lock) {
        this->lock = lock;
        if (this->lock)
            cout << "PickUpPlace: <lock>₩n";
        else
            cout << "PickUpPlace: <unlock>₩n";
    }
    bool isLock() {
        return this->lock;
    }
    void setEmpty(bool empty) {
        this->empty = empty;

        if (empty)
            setState("_____");
    }
    bool isEmpty() {
        return this->empty;
    }
    void setState(string newState) {
        this->state = newState;
        showState();
    }
    void showState() { cout << "PickUpPlace [" << state << "]₩n"; }
};

#endif
```