**2018 전공기초프로젝트2 (2040) 화금반**

**[1차 설계 문서]**

**: (6조) 음료 자판기 시뮬레이션 프로그램**

****

조장 201511206 왕윤성

201511197 방승희

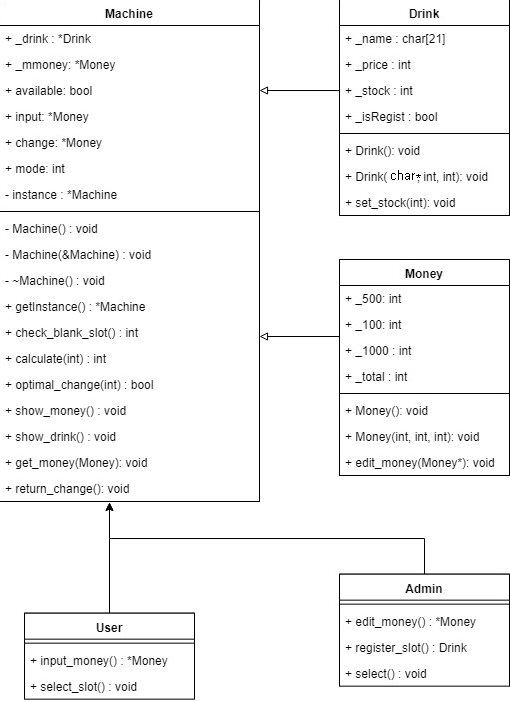
201611186 김나경

제출일 : 2018년 09월 30일

[순서]

* 언어 : c++ 사용 객체지향형 / 툴 : Visual Studio 2017
* .txt 파일에 vending machine 정보 저장
* UI 환경 : CUI(Command Line User Interface)
* 객체 정의
* 순서도
* 파일 읽고 쓰기의 모듈화
* 예외처리의 모듈화

[객체 클래스 설계도]



# 음료 **Drink**

* + 멤버 변수
    - **+ bool \_isRegist** : 등록한 슬롯인지 판단
    - **+ char \_name[21]** : 음료의 이름 저장(20자까지)
    - **+ int \_price** : 음료의 가격 저장
    - **+ int \_stock** : 음료의 재고 저장
  + 멤버 함수
    - **+ void Drink()** : 생성자. 멤버 변수들을 초기화 함.

(\_isRegist, \_name, \_price, \_stock) =

(false, null, 0, 0)

* + - **+ void Drink(char\* name, int price, int stock)** : 생성자 오버로딩.

(\_isRegist, \_name, \_price, \_stock) =

(true, name, price, stock)

* + - **+ void set\_stock(int stock)** : 음료의 재고를 파라미터 stock으로

변경함. 추가 혹은 제거할 양만큼 입력이

아니라 변경할 양을 입력 받는 것.

# 돈 **Money**

* + 멤버 변수
    - **+ int \_100** : 100원의 개수
    - **+ int \_500** : 500원의 개수
    - **+ int \_1000** : 1000원의 개수
    - **+ int \_total** : 총액
  + 멤버 함수
    - **+ void Money()** : 생성자. 멤버 변수들을 초기화 함.

(\_100, \_500, \_1000, \_total) = (0, 0, 0, 0)

* + - **+ void Money(int input\_100, int input\_500, int input\_1000)** :

생성자 오버로딩.

(\_100, \_500, \_1000, \_total) =

(input\_100, input\_500, input\_1000, total계산)

* + - **+ void edit\_money(\*Money income)** : 금고의 잔고를 파라미터 income을 더해 갱신함.

# 자판기 **Machine**

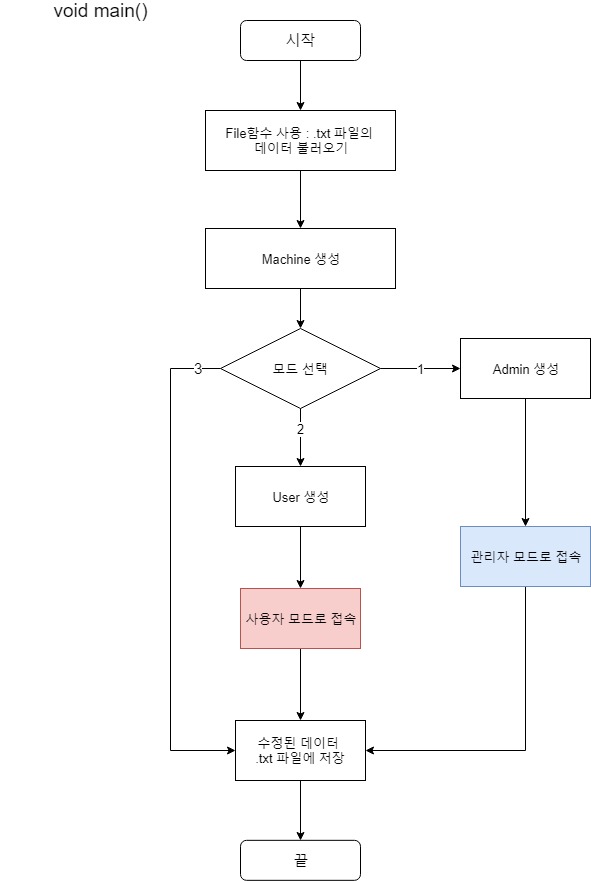
* + 싱글톤 패턴
    - 프로그램 전체에서 딱 한 번만 생성되고, 재 생성되지 않아야 함.
    - 재 생성시 처음에 만들어 놓은 자판기를 참조하게 해야함.
  + 멤버 변수
    - **+ \*Machine instance** : 메인함수에서 처음으로 생성한 Machine을 가리키는 객체포인터.
    - **+ Drink \_drink** : 자판기의 음료 저장.
    - **+ \*Money \_mmoney** : 자판기가 보유한 돈 저장.
    - **+ bool available** : 재고 및 잔고 이용 가능 여부 저장.(true : 사용가능, false : 재고 없음 또는 잔돈 없음)
    - **+ \*Money input** : 투입한 금액 저장
    - **+ \*Money change** : 잔돈 저장. 최적 반환계산을 통해 나온 결과를 저장.
    - **+ int mode** : 관리자, 사용자 모드 구분 값 저장. ( 1 : 관리자, 2 : 사용자, 3 : 종료)
  + 멤버 함수
    - **- void Machine()** : 생성자. 멤버 변수들을 초기화함.
    - **- void Machine(&Machine)** : 생성자 오버로딩.
    - **- void ~Machine()** : 소멸자. 데이터 해제.
    - **+ \*Machine getInstance()** : 싱글톤패턴 구현. 멤버변수 instance가 이미 있다면 그 instance를 리턴하고, 없다면(처음 생성할 경우) 새로 생성해서 리턴함.
    - **+ int check\_blank\_slot()** : drink[10]에서 인덱스 순으로 했을 때 처음으로 찾은 빈 슬롯의 인덱스 번호 리턴.
    - **+ int calculate(int slot\_num)** : 슬롯 번호를 받아서 해당 슬롯의 음료를 투입된 금액으로부터 뺌(차액 계산). 돌려줄 금액 구하는 함수.
    - **+ bool optimal\_change(int tmp)** : Machine의 금고(\_mmoney)를 확인해 잔돈을 반환할 수 있는지 여부 확인. 동시에 최적으로 반환할 수 있는 방법 계산. (최적 반환이란 기계에 주어진 돈 내에서 큰 단위의 돈을 최대한 많이 쓰는 방법으로 반환하는 것.) 돌려줄 돈(change)를 계산하는 것.
    - **+ void show\_money()** : 금고에 있는 단위별 돈의 개수와 전체 금액을 보여줌.
    - **+ void show\_drink()** : 슬롯에 저장된 음료를 보여줌. 멤버 변수 mode의 값에 따라 1(관리자)이면 재고까지 포함해서 보여줌. 2(사용자)이면 재고를 뺀 음료 슬롯번호, 음료이름, 가격 까지만 보여줌.
    - **+ get\_money(Money\* m)** : 투입된 돈(m)을 금고(\_mmoney)에 저장.
    - **+ void retrun\_change()** : 잔돈 반환. 사용자의 주문을 받고 일어날 수 있는 각종 오류사항(미등록 슬롯, 재고 없음, 투입금액 부족, 기계내 잔돈 부족)이나 정확한 주문 후 받았던 돈을 다시 돌려줄 때 이 함수를 호출함. 금고(\_mmoney)로부터 돌려줄 돈(change)를 뺌. 투입 받은 금액과 돌려줄 금액은 다시 0으로 초기화

# 사용자 **User**

* + 멤버 함수
    - **+ \*Money input\_money()** : 사용자의 돈을 투입받음. 투입받은 돈은 Money 형식으로 리턴함. Money형식으로 리턴시 각 돈의 종류마다 몇개인지의 정보를 함께 넘길 수 있음.
    - **+ void select\_slot()** : 사용자로부터 슬롯을 입력 받음.(주문 받기)

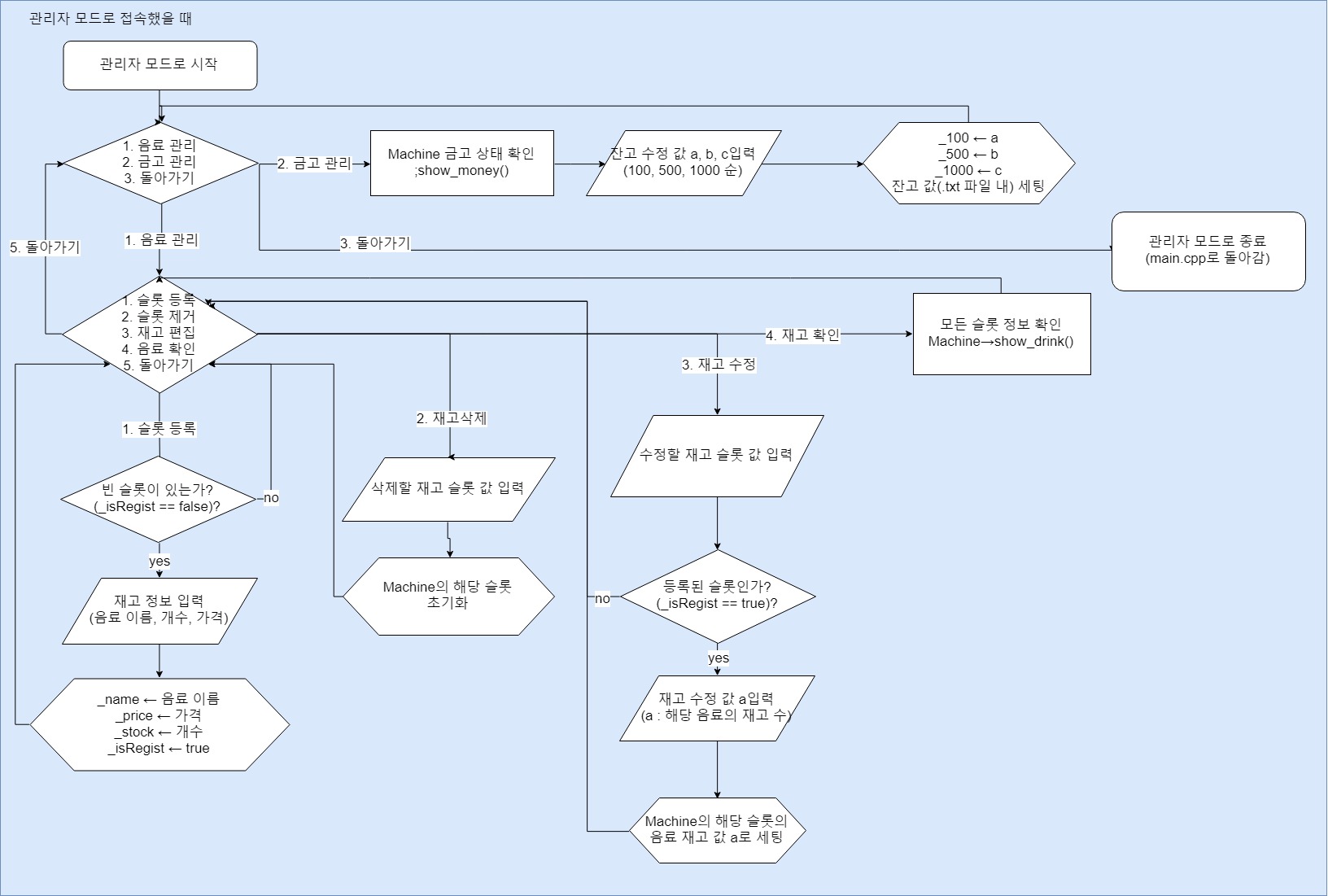
# 관리자 **Admin**

* + 멤버 함수
    - **+ \*Money edit\_money()** : 관리자가 기계의 금고를 변경하고자 할때 사용하는 함수. 추가하고 싶다면 양의 정수를, 제거하고 싶다면 음의 정수를 입력함. 추가/제거해서 저장할 숫자를 입력하는 것이아니라 추가/제거하고 싶은 만큼을 입력하는 것. 해당 정보는 \*Money로 리턴됨.
    - **+ Drink register\_slot()** : 관리자가 기계의 음료 슬롯을 등록한다. 이름과 가격, 재고를 입력한함. 해당 정보는 \*Drink로 리턴됨.
    - **+ void select()** : 관리자로부터 변경할 음료 슬롯 번호나 재고 수를 입력받을 때 호출하는 함수.

[순서도]

* 옳은 입력이 들어올 때까지 while로 과정을 반복한다.

# [관리자 모드]



* 세가지 메뉴를 입력 받는다.
* 1. 음료관리, 2. 금고관리, 3. 돌아가기가 있고 그 외의 입력은 다시 받는다.
* 음료 관리에서는 다시 네가지 메뉴를 입력 받는다.
* 1. 슬롯 등록, 2. 슬롯 제거, 3. 재고 편집, 4. 음료 확인, 5. 돌아가기가 있고 그 외의 입력은 다시 받는다.
  + 1. 슬롯 등록시 Machine->check\_blank\_slot()으로 인덱스순으로 맨 처음으로 발견되는 빈 슬롯을 찾는다. 없다면 관리자 모드 종료.

찾은 후 Admin->regist\_slot() - Drink(name, price, stock)입력

Machine의 해당 drink[index] 갱신.

* + 2. 슬롯 제거 시 Admin->select() 제거할 슬롯 번호 입력

Machine의 해당 drink[index] 초기화

* + 3. 재고 편집 시 Admin->select() 편집할 슬롯 번호 입력

등록된 슬롯인지 검사 후 등록됐다면 Admin->select()로 변경할 재고 수 입력

Machine의 해당 drink[index] 갱신

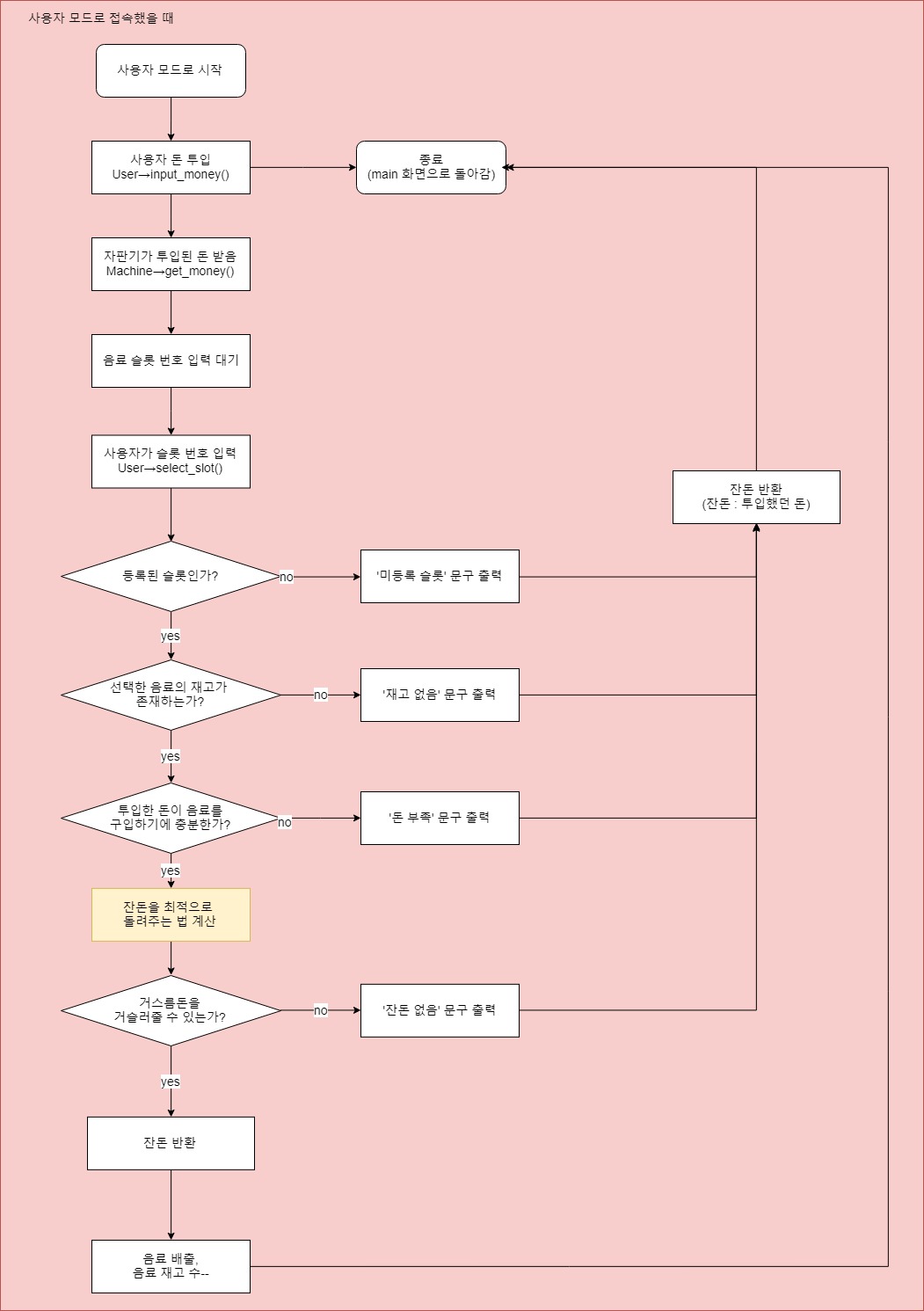
등록 안됐다면 메뉴로 돌아가기

* + 4. 음료 확인 시

Machine->show\_drink()

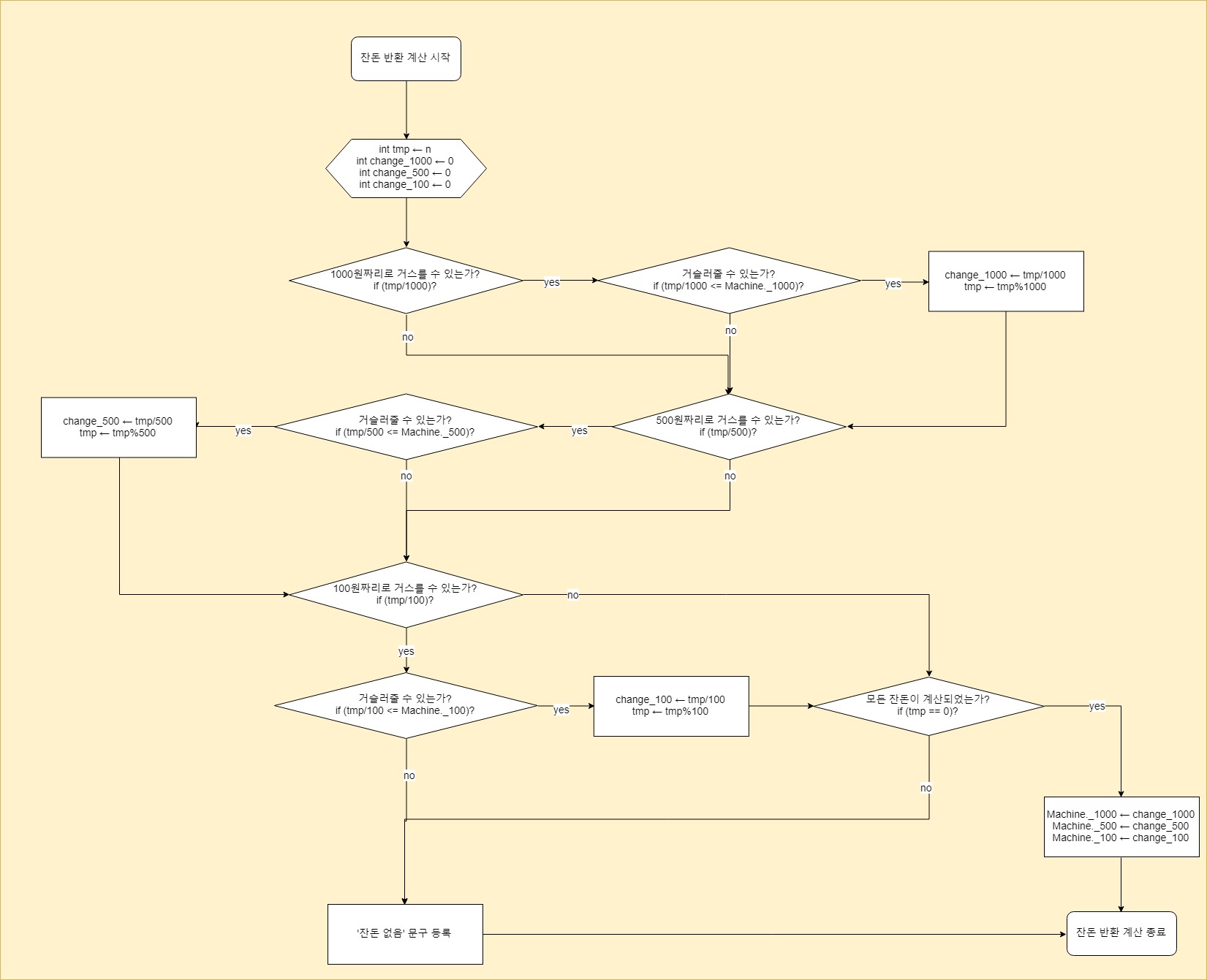
* 금고관리에서는 별도의 메뉴를 입력받지 않고, 금고 확인과 금고 편집이 동시에 이루어진다.
  + 금고 확인 시 Machine->show\_money()
  + 금고 편집 시 Admin->edit\_money()

# [사용자 모드]



* Machine->show\_drink()로 음료 보여주기
* User->input\_money()
* Machine->get\_money()
* User->select\_slot()
* 슬롯의 등록된 슬롯인지, 슬롯에 재고가 있는지, 투입 금액이 가격 이상인지 체크
* 이 모든 체크를 넘기지 못하면 받은 돈을 돌려주고(Machine->return\_change()) 모드 선택화면으로 돌아감
* Machine->calculate()
* Machine->optimal\_change()가 return false를 하면 기계 내 잔돈 없음이므로 받은 돈을 돌려주고(Machine->return\_change()) 모드 선택화면으로 돌아감
* Machine->optimal\_change()가 return true를 하면 optimal\_change로 최적의 방법으로 돈을 돌려주고(Machine->return\_change()), 음료 재고를 하나 줄임.

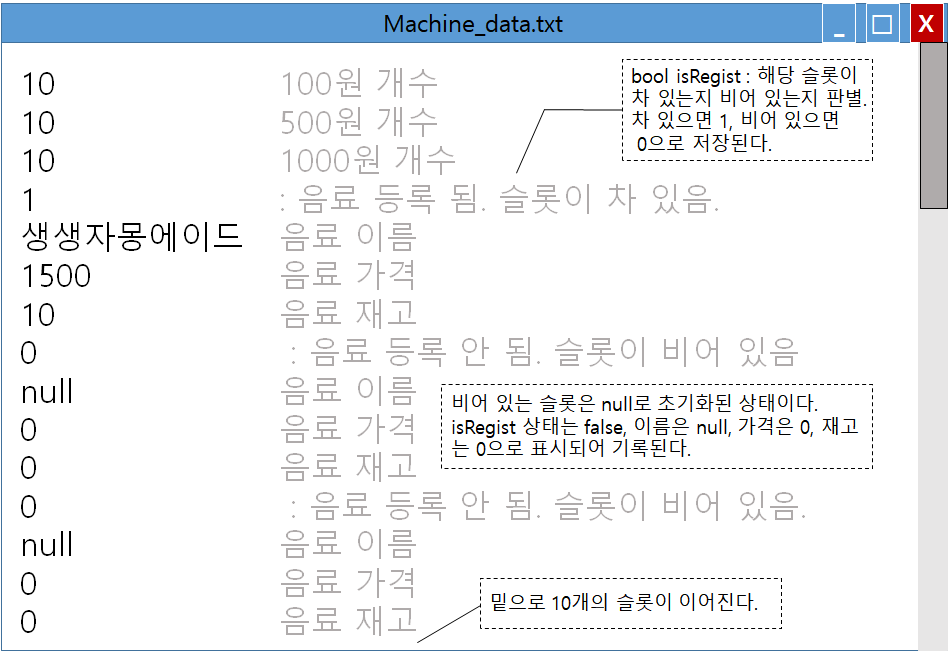
# [잔돈 반환] Machine->bool optimal\_change(int tmp)



* 입력으로 잔돈의 총액을 tmp 변수에 받음
* Machine->change를 초기화함
* change\_100, change\_500, change\_1000을 각각 0으로 초기화함
* 제일 큰 단위인 1000원으로 tmp를 나누어서 1000원으로 나눠 줄 수 있는 가장 큰 개수를 구함.
* 그 개수를 change\_1000원에 저장
* change\_1000이 Machine->\_1000 이하 인지 확인
* 이하라면 tmp를 1000원으로 바꾼 만큼의 금액을 뺀 금액으로 갱신
* 그렇지 않다면 tmp를 따로 갱신하지 않고, change\_1000을 다시 0으로 초기화.
* 이 과정을 다음단위 500, 100원으로 반복
* 마지막으로 change\_100이 Machine->\_100보다 작으면 반환 가능하므로 machine->change를 new Money(change\_100, change\_500, change\_1000)로 갱신 후 return true
* 그보다 크다면 반환할 수 없으므로 return false

[파일 읽고 쓰기의 모듈화]

* + Machine\_data.txt에 데이터 저장.



* + 메인함수의 처음에서 데이터를 읽어와서 프로그램 객체에 저장하고,
  + 메인함수의 끝에서 (즉, 한 사용자 혹은 관리자가 필요한 기능을 다 수행한 후) 프로그램 객체로 부터 데이터를 텍스트 파일에 변경사항 저장

[예외처리의 모듈화]

* + 예외처리 할 것들
    - 기계의 금고에서
      * 0개 <=100원짜리 개수 =<100개
      * 0개 <=500원짜리 개수 =<50개
      * 0장 <=1000원짜리 장수 =<50장
    - 음료 가격에서
      * 최대 2000원으로 설정 가능
      * 100원 단위로 설정 가능
    - 음료 이름에서
      * 글자 수는 영어 기준으로 20자
    - 음료 슬롯의 수는 10개로 고정
    - 음료 슬롯당 재고는 최대 15개
    - 메뉴 선택 시, 메뉴 이외의 번호나 문자를 입력했을 경우
  + **bool exception\_test(int num, int max, int min)**

: 최대와 최소를 검사하는 함수.

: int형 변수의 예외처리를 하는 함수이다. 매개변수로 ‘**num** : 검사할 변수’, ‘**max** : 최대값’, ‘**min** : 최소값’을 차례로 입력받아야 한다. 만약 변수가 정수형이 아니거나, 지정한 범위 [min, max] 미만이거나 초과이면 false값을 리턴한다. 검사할 변수가 옳은 형식이라면 true 값을 리턴한다.

* + **메뉴 선택 시 메뉴 이외의 번호나 문자를 입력했을 경우 (해당 화면에서 처리)**

: switch문을 이용하여, 지정한 번호나 문자 외에는 default의 경우로 처리해 해결한다.

* + **음료 이름이 20자를 초과했을 경우 (해당 화면에서 처리)**

: 음료의 이름을 받는 함수로 getline을 이용하여 처리한다. getline()은 매개변수로 char\*형 변수1, int형 변수2를 이용하는데, 변수1의 글자 수는 변수2를 넘지 못한다. 그러므로 변수2에 (int)를 사용해 20자가 넘으면 자동으로 무시되게 하여 처리할 수 있다. 입력을 받은 후 버퍼를 지워주고 ignore를 사용하여 int의 최댓값(2147483647)을 넘지 않는 모든 입력에 대해 입력이 가능하다.