MARKET MANAGEMENT APPLICATION

20185168 Gie Tae, Lee
20183622 Tae Woo, Kim
20185785 Sang Yun, Lee
20184165 Tae Un, Lee
20185949 Ye Seul, Choi

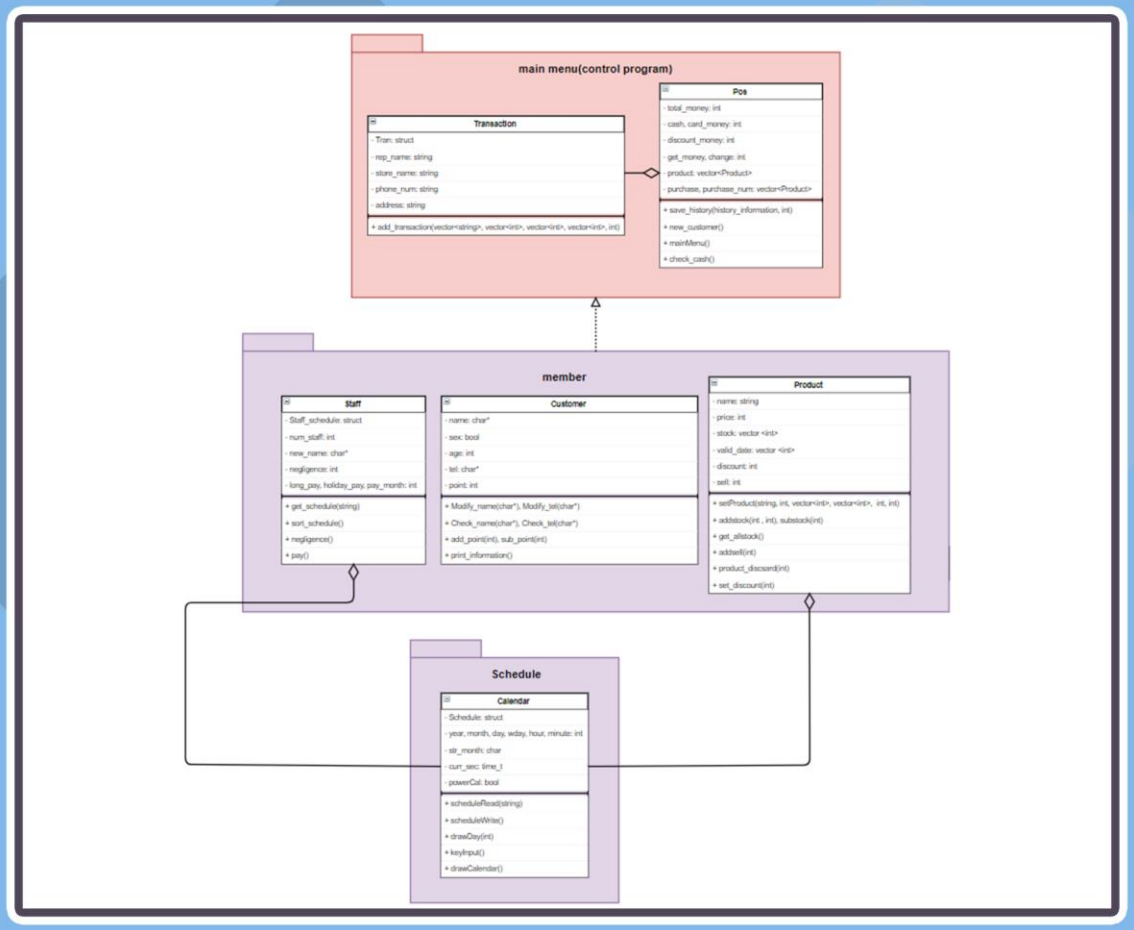
INDEX

1. Goal of design
2. UML
3. Classes
   - 3-1. Calendar
   - 3-2. Product
   - 3-3. Staff
   - 3-4. POS
   - 3-5. Transaction
   - 3-6. Customer
4. Features

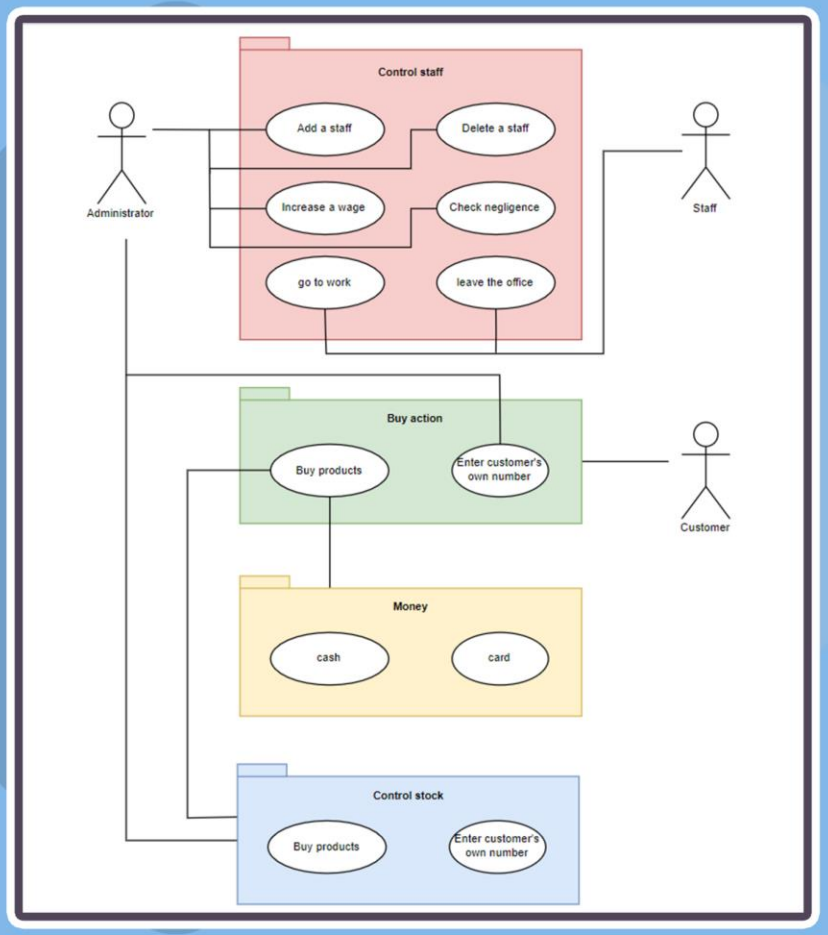
Goal of design

- It was important to create a realistic and easy to use MMA.
- Feeling constrained by the Visual Studio program, use a tool called QT.
- Divided into 6 classes, we tried to make as many functions as possible.
- By putting special functions, we made a distiction between real MMA.

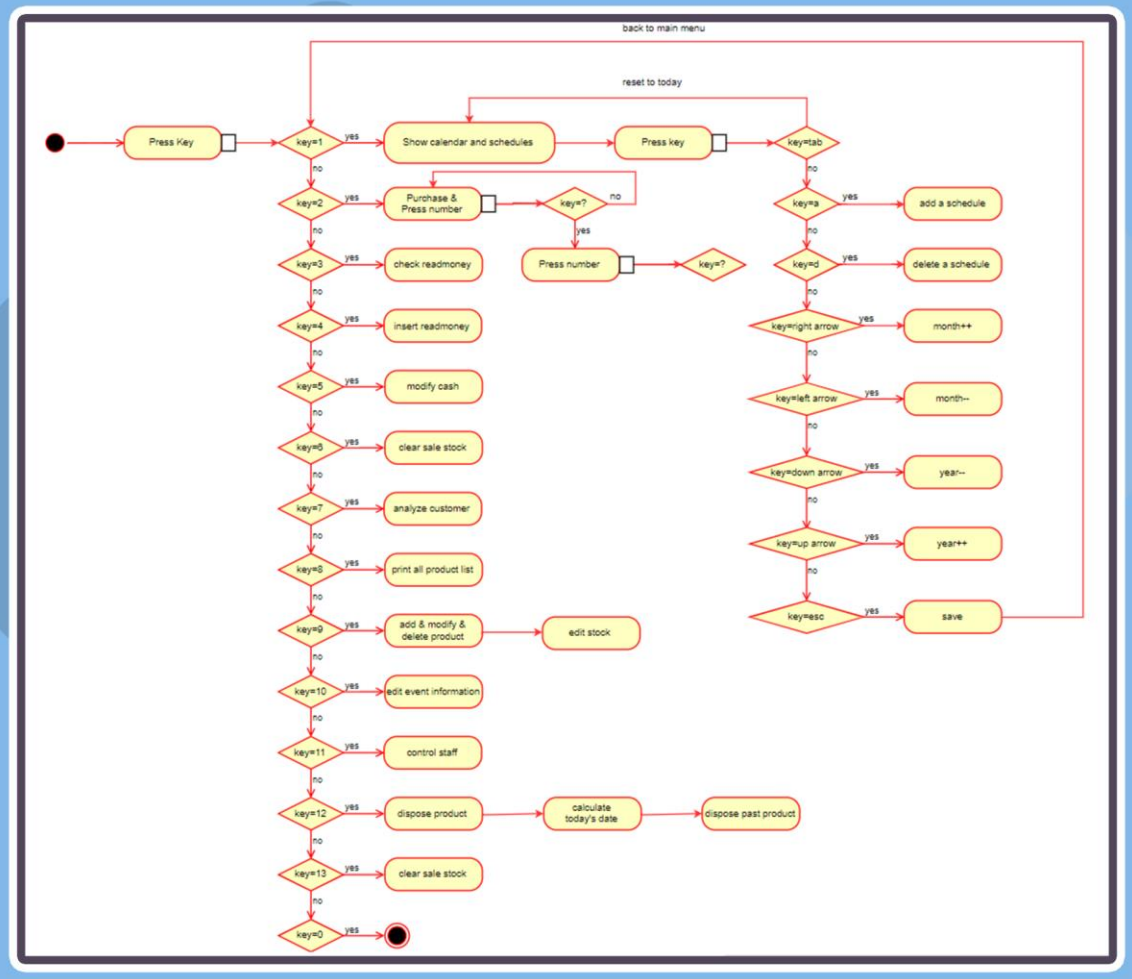
UML


main menu(control program)
Pos
- total_money: int
- cash, card_money: int
- discount_money: int
- get_money, change: int
- product: vector<Product>
- purchase, purchase_num: vector<Product>
+ save_history(history_information, int)
+ new_customer()
+ mainMenu()
+ check_cash()
Transaction
- Tran: struct
- rep_name: string
- store_name: string
- phone_num: string
- address: string
+ add_transaction(vector<string>, vector<int>, vector<int>, vector<int>, int)
member
Staff
- Staff_schedule: struct
- num_staff: int
- new_name: char*
- negligence: int
- long_pay, holiday_pay, pay_month: int
+ get_schedule(string)
+ sort_schedule()
+ negligence()
+ pay()
Customer
- name: char*
- sex: bool
- age: int
- tel: char*
- point: int
+ Modify_name(char*), Modify_tel(char*)
+ Check_name(char*), Check_tel(char*)
+ add_point(int), sub_point(int)
+ print_information()
Product
- name: string
- price: int
- stock: vector <int>
- valid_date: vector <int>
- discount: int
- sell: int
+ setProduct(string, int, vector<int>, vector<int>, int, int)
+ addstock(int , int), substock(int)
+ get_allstock()
+ addsell(int)
+ product_discsend(int)
+ set_discount(int)
Schedule
Calendar
- Schedule: struct
- year, month, day, wday, hour, minute: int
- str_month: char
- curr_sec: time_t
- powerCal: bool
+ scheduleRead(string)
+ scheduleWrite()
+ drawDay(int)
+ keyInput()
+ drawCalendar()

UML


Control staff
Add a staff
Delete a staff
Increase a wage
Check negligence
go to work
leave the office
Administrator
Staff
Buy action
Buy products
Enter customer's own number
Customer
Money
cash
card
Control stock
Buy products
Enter customer's own number

UML


back to main menu
reset to today
Press Key
key=1
yes
Show calendar and schedules
Press key
key=tab
no
key=2
yes
Purchase & Press number
key=?
no
yes
key=a
yes
add a schedule
no
key=3
yes
check readmoney
Press number
key=?
key=d
yes
delete a schedule
no
key=4
yes
insert readmoney
key=right arrow
yes
month++
no
key=5
yes
modify cash
key=left arrow
yes
month--
no
key=6
yes
clear sale stock
key=down arrow
yes
year--
no
key=7
yes
analyze customer
key=up arrow
yes
year++
no
key=8
yes
print all product list
key=esc
yes
save
no
key=9
yes
add & modify & delete product
edit stock
no
key=10
yes
edit event information
no
key=11
yes
control staff
no
key=12
yes
dispose product
calculate today's date
dispose past product
no
key=13
yes
clear sale stock
no
key=0
yes

3-1. Calendar

void scheduleRead(string filename)

: A method for reading text files on a holiday or event schedule, such as Christmas or Pepero Day.

void scheduleWrite()

: A method for writing staffs schedules.

int D_day(int d_year, int d_month, int d_day)

: A method that calculates d-day from Jan,1st 2018.

Vector<int> returnTime()

: A method that returns current time by vector<int>.

3-1. Calendar

void scheduleRead(string filename)

: A method for reading text files on a holiday or event schedule, such as Christmas or Pepero Day.

```
void Calendar::scheduleRead(string filename) {

    ifstream inFile(filename);

    while (!inFile.eof()) {
        char inputLine[100] = { 0, };
        inFile.getline(inputLine, 100);

        if (inputLine[0] == NULL) { break; }

        string r_type;
        char* r_note;

        int r_year, r_month, r_day, r_start_hour, r_start_min, r_end_hour, r_end_min;

        r_type = strtok(inputLine, "/");
        r_year = atoi(strtok(NULL, "/")) - 2019;
        r_month = atoi(strtok(NULL, "/"));
        r_day = atoi(strtok(NULL, "/"));

        schedule[num_sch[r_year][r_month]][r_year][r_month].sch_type = r_type;
        schedule[num_sch[r_year][r_month]][r_year][r_month].sch_day = r_day;
```

```
        if (r_type[0] == 'a') {
            r_start_hour = atoi(strtok(NULL, "/"));
            schedule[num_sch[r_year][r_month]][r_year][r_month].sch_start_hour = r_start_hour;


            r_start_min = atoi(strtok(NULL, "/"));
            schedule[num_sch[r_year][r_month]][r_year][r_month].sch_start_min = r_start_min;

            r_end_hour = atoi(strtok(NULL, "/"));
            schedule[num_sch[r_year][r_month]][r_year][r_month].sch_end_hour = r_end_hour;

            r_end_min = atoi(strtok(NULL, "/"));
            schedule[num_sch[r_year][r_month]][r_year][r_month].sch_end_min = r_end_min;

            r_note = strtok(NULL, "/");
            strcpy(schedule[num_sch[r_year][r_month]][r_year][r_month].sch_note, r_note);
            if(r_note == NULL)
                schedule[num_sch[r_year][r_month]][r_year][r_month].sch_note ==NULL;
        }
        else if (r_type[0] == 'e') {

        }
        else {      // b, c, d
            r_note = strtok(NULL, "/");
            strcpy(schedule[num_sch[r_year][r_month]][r_year][r_month].sch_note, r_note);
        }
        num_sch[r_year][r_month]++;
    }


    //sort(student, student + numStudent, compare);

    inFile.close();
```

3-1. Calendar

void scheduleWrite()

: A method for writing staffs schedules.

```
void Calendar::scheduleWrite() {    ////////////////////////////////////// a,e 나눠야 할수도?????????

    ofstream outFile("staff_schedule.txt");
    for (int y = 0; y < 10; y++) {
        for (int m = 1; m < 13; m++) {
            for (int i = 0; i < num_sch[y][m]; i++) {

                if (schedule[i][y][m].sch_type[0] == 'a') {
                    outFile << "a/" << y + 2019 << "/" << m << "/" << schedule[i][y][m].sch_day << "/";
                    outFile << schedule[i][y][m].sch_start_hour << "/" << schedule[i][y][m].sch_start_min << "/";
                    outFile << schedule[i][y][m].sch_end_hour << "/" << schedule[i][y][m].sch_end_min << "/";
                    outFile << schedule[i][y][m].sch_note << endl;
                }
                else {          // e일경우

                }
            }
        }
    }

    outFile.close();
}
```

3-1. Calendar

int D_day(int d_year, int d_month, int d_day)

: A method that calculates d-day from Jan,1st 2018.

```
int Calendar::D_day(int d_year, int d_month, int d_day) {
    int anni_all_day = 0, ye = 0, mo = 0, a = 0;
    int now_all_day = 0, ye2 = 0, mo2 = 0, b = 0;
    int D_day = 0;
    for (int i = 2018; i < d_year; ++i) {
        ye += 365;
        if (dayMonthLimit(2) == 29)
            ye += 366;
    }
    for (int i = 0; i < d_month - 1; ++i) {
        mo += dayMonthLimit(i);
    }
    if (dayMonthLimit(year) == 29 && d_month > 2) {
        a = 1;
    }
    anni_all_day = d_day + ye + mo + a;

    for (int i = 2018; i < year; ++i) {
        ye2 += 365;
        if (dayMonthLimit(2) == 29)
            ye2 += 366;
    }
    for (int i = 0; i < month - 1; ++i) {
        mo2 += dayMonthLimit(i);
    }
    if (dayMonthLimit(year) == 29 && month > 2) {
        b = 1;
    }
    now_all_day = day + ye2 + mo2 + b;
    D_day = (anni_all_day - now_all_day);
    return D_day;
}
```

3-1. Calendar

Vector<int> returnTime()

: A method that returns current time by vector<int>.

```
vector<int> Calendar::returnTime() {
    vector<int> clock = { year, month, day, hour, minute };
    return clock;
}
```

3-2. Product

void setProduct(string name, int price, vector<int> valid_date, vector<int> stock, int discount, int sell)

: A method for adding product with several parameters.

void addstock(int num, int date)

: A method for adding stock with date

int substock(int n)

: A method that decreases stock when selled n products.

void set_discount(int n)

: A method that modify product's events.

3-2. Product

void setProduct(string name, int price, vector<int> valid_date, vector<int> stock, int discount, int sell)

: A method for adding product with several parameters.

```
void Product::setProduct(string name, int price, vector<int> valid_date, vector<int> stock, int discount, int sell) {
    this->setname(name);
    this->setprice(price);
    for (int i = 0; i < valid_date.size(); i++) {
        this->setvalid_date(valid_date.at(i));
        this->setstock(stock.at(i));
    }
    this->setdiscount(discount);
    this->setsell(sell);
    //this->setcategory(category);
} //배열에 등록
```

3-2. Product

void addstock(int num, int date)

: A method for adding stock with date

```
void Product::addstock(int num , int date) {
    this->stock.push_back(num);
    this->valid_date.push_back(date);
        :
}//재고 추가
```

3-2. Product

int substock(int n)

: A method that decreases stock when selled n products.

```
int Product::substock(int n) {
    int sum = 0;
    for (int i = 0; i < this->stock.size(); i++) {
        sum += stock[i];
    }
    if (sum < n) {
        //재고가 부족합니다 출력
    }
    else {
        while (n > 0) {
            int temp = n;
            n -= this->stock[0];
            this->stock[0] -= temp;

            if (this->stock[0] <= 0) {
                this->stock.erase(stock.begin());
                this->valid_date.erase(valid_date.begin());
            }
        }
    }
    return 1;////////////////////////////////////////////////////////////////수정
}//재고 감소
```

3-2. Product

void set_discount(int n)

: A method that modify product's events.

```
void Product::set_discount(int n) {
    this->discount = n;
}//물품 행사 추가
```

3-3. Staff

void set_work_start(int staff_num, int schedule_num)

: A method for setting staff's start time.

void set_work_end(int staff_num, int schedule_num)

: A method for setting staff's end time.

void set_negligence()

: A method that checks staff's negligences.

void calculate_work_time()

: A method that calculates staff's work time.

3-3. Staff

void sort_schedule()
: A method that sorts staff's schedules by using sort algorithm.

void upgrade_pay_hour()
: A method that increases pay for no negligence staff.

void pay()
: A method for pay to staffs.

3-3. Staff

void set_work_start(int staff_num, int schedule_num)

: A method for setting staff's start time.

```
//출근 시간 set
void Staff::set_work_start(int staff_num, int schedule_num) {
    staff2[staff_num].sch2[schedule_num].cur_work_start_time = returnTime();
}
```

3-3. Staff

void set_work_end(int staff_num, int schedule_num)

: A method for setting staff's end time.

```
//퇴근시간 set
void Staff::set_work_end(int staff_num, int schedule_num) {
    staff2[staff_num].sch2[schedule_num].cur_work_end_time = returnTime();
}
```

3-3. Staff

void set_negligence()

: A method that checks staff's negligences.

```
//근무 태만 횟수 set
void Staff::set_negligence() {
    for(int i=0;num_staff;i++)
        for (int j = 0; j < 30; j++) {
            //조기퇴근
            if ((staff2[i].sch2[j].work_end_hour - staff2[i].sch2[j].cur_work_end_time.at(4) > 0) || ((st
                staff2[i].negligence++;
            //지각
            if ((staff2[i].sch2[j].cur_work_start_time.at(4) - staff2[i].sch2[j].work_start_hour > 0) ||
                staff2[i].negligence++;
        }
}
```

3-3. Staff

void calculate_work_time()

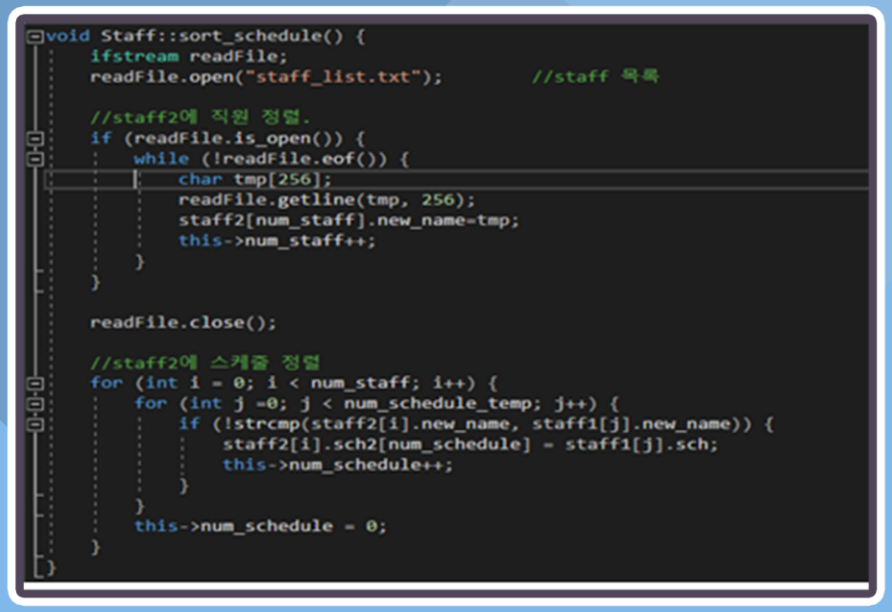
: A method that calculates staff's work time.

```
void Staff::calculate_work_time() { //해당스태프번호, 해당스태프의스케줄번호
    for (int staff_num = 0; staff_num < num_staff; staff_num++) {
        for (int schedule_num = 0; schedule_num < 30; schedule_num++) {
            //스태프 해당 스케줄에 일한 시간 계산
            staff2[staff_num].sch2[schedule_num].work_time_day = ((staff2[staff_num].sch2[schedule_num].w

            if (staff2[staff_num].sch2[schedule_num].work_time_day >= 8)    //8시간 이상 일하면
                staff2[staff_num].sch2[schedule_num].work_time_day = -1;    //1시간은 휴식시간

            else if (staff2[staff_num].sch2[schedule_num].work_time_day >= 4)   //4시간 이상 8시간 미만 일
                staff2[staff_num].sch2[schedule_num].work_time_day = -0.5;  //30분은 휴식시간
        }
    }
}
```

3-3. Staff

void sort_schedule()

: A method that sorts staff's schedules by using sort algorithm.



```
void Staff::sort_schedule() {
    ifstream readFile;
    readFile.open("staff_list.txt");        //staff 목록

    //staff2에 직원 정렬.
    if (readFile.is_open()) {
        while (!readFile.eof()) {
            char tmp[256];
            readFile.getline(tmp, 256);
            staff2[num_staff].new_name=tmp;
            this->num_staff++;
        }
    }

    readFile.close();

    //staff2에 스케줄 정렬
    for (int i = 0; i < num_staff; i++) {
        for (int j =0; j < num_schedule_temp; j++) {
            if (!strcmp(staff2[i].new_name, staff1[j].new_name)) {
                staff2[i].sch2[num_schedule] = staff1[j].sch;
                this->num_schedule++;
            }
        }
        this->num_schedule = 0;
    }
}
```

3-3. Staff

void upgrade_pay_hour()

: A method that increases pay for no negligence staff.

```
void Staff::upgrade_pay_hour() {
    for (int i = 0; i < num_staff; i++)
        if(staff2[i].negligence==0)          //negligence 횟수가 없다면
            staff2[i].pay_hour += 200;       //시급 200원 인상
}
```

3-3. Staff

void pay()

: A method for pay to staffs.

```
Staff::pay() {
ofstream file;
file.open("staff_pay.txt", ios::app);       //staff pay 목록

for (int i = 0; i < num_staff; i++) {

    if (num_staff > 5) {                    // 직원이 5인 이상이면 연장수당, 휴일수당
        if (int j = 0; j < 20; j++) {

            //하루에 일한 시간이 8시간 초과면 연장수당
            //연장수당은 기본 시급의 0.5배
            if (staff2[i].sch2[j].work_day > 8) {
                staff2[i].long_pay += (int)((staff2[i].sch2[j].work_day - 8)*staff2[i].pay_hour*0.5);
            }

            //휴일수당은 휴일에 8시간 이하로 근무하면 기본 시급의 0.5배
            //휴일수당은 휴일에 8시간 초과하여 근무하면 기본 시급의 1배
            if (staff2[i].sch2[j].cur_work_start_time.at(3) == 0) {     //휴일에 일하였다면
                if (staff2[i].sch2[j].work_time_day > 8)
                    staff2[i].holiday_pay += (int)((staff2[i].sch2[j].work_day - 8)*staff2[i].pay_hour*1.
                else
                    staff2[i].holiday_pay += (int)((staff2[i].sch2[j].work_day - 8)*staff2[i].pay_hour*0.
            }
        }

        //월급 계산
        staff2[i].pay_month += long_pay + holiday_pay + (staff2[i].sch2[j].work_day)*staff2[i].pay_hour;

        file << staff2[i].new_name << "님의 연장수당은 " << staff2[i].long_pay << "원입니다." << endl;     /
        file << staff2[i].new_name << "님의 휴일수당은 " << staff2[i].holiday_pay << "원입니다." << endl; /
        file << staff2[i].new_name << "님의 월급은 " << staff2[i].pay_month << "원입니다." << endl;       /

    }
}


file.close();
```

3-4. POS 

void productRead(const char* filename)
: A method that reads text file which has product list.

void productWrite(const char* filename)
: A method that writes text file which has product list.

void set_total_money()
: A method for setting money in device.

void input_total_cash()
: A method for writting money in device.

3-4. POS

void modify_product()
: A method that modifies products.

void add_stock()
: A method that adds product's stock.

void sub_stock()
: A method for delete product which is not long before the expiration date.

void input_buy_product()
: A method that implements payment process.

3-4. POS 

void calculate_cash_buy(int q)
: A method that calculates exchange after payment.

void discount_string(int dis)
: A method that sets product's events.

void trash_product_clear()
: A method for disposes products that are out of date.

3-4. POS

void productRead(const char* filename)

: A method that reads text file which has product list.

```
ifstream inFile(filename);

while (!inFile.eof()) {
    char inputLine[200] = { 0, };
    inFile.getline(inputLine, 200);

    if (inputLine[0] == NULL) { break; }
    string r_name;
    int r_price, r_discount, r_sell, r_category;
    //int r_valid_date;
    //int r_loacation;

    r_name = strtok(inputLine, "/");
    r_price = atoi(strtok(NULL, "/"));
    char* r_stock_index = strtok(NULL, "/");
    r_discount = atoi(strtok(NULL, "/"));
    r_sell = atoi(strtok(NULL, "/"));

    vector <int> r_valid_date;
    vector <int> r_stock;
    int r_index;
    char* r_c;
```

```
    bool c = true;
    while (1) {
        if (c) {
            r_index = atoi(strtok(r_stock_index, ".#"));
            c = false;
        }
        else {
            r_c = strtok(NULL, ".#");
            if (r_c == NULL)
                break;
            r_index = stoi(r_c);
        }

        r_valid_date.push_back(r_index);
        r_index = atoi(strtok(NULL, ".#"));
        r_stock.push_back(r_index);
    }


    Product input_p;
    input_p.setProduct(r_name, r_price, r_valid_date, r_stock, r_discount, r_sell);
    product.push_back(input_p);
}
```

3-4. POS

void productWrite(const char* filename)

: A method that writes text file which has product list.

```
void POS::productWrite(const char* filename) {
    ofstream outFile(filename);

    for (int i = 0; i < product.size(); i++) {
        outFile << product.at(i).getname() << "/" << product.at(i).getprice() << "/";
        for (int j = 0; j < product.at(i).getstock().size(); j++) {
            outFile << product.at(i).getvalid_date().at(j) << "." << product.at(i).getstock().at(j);
            if (j == product.at(i).getstock().size() - 1)
                outFile << "##/";
            else
                outFile << ".";
        }
        outFile << product.at(i).getdiscount() << "/" << product.at(i).getsell() << endl;
    }
    outFile.close();

}
```

3-4. POS

void set_total_money()

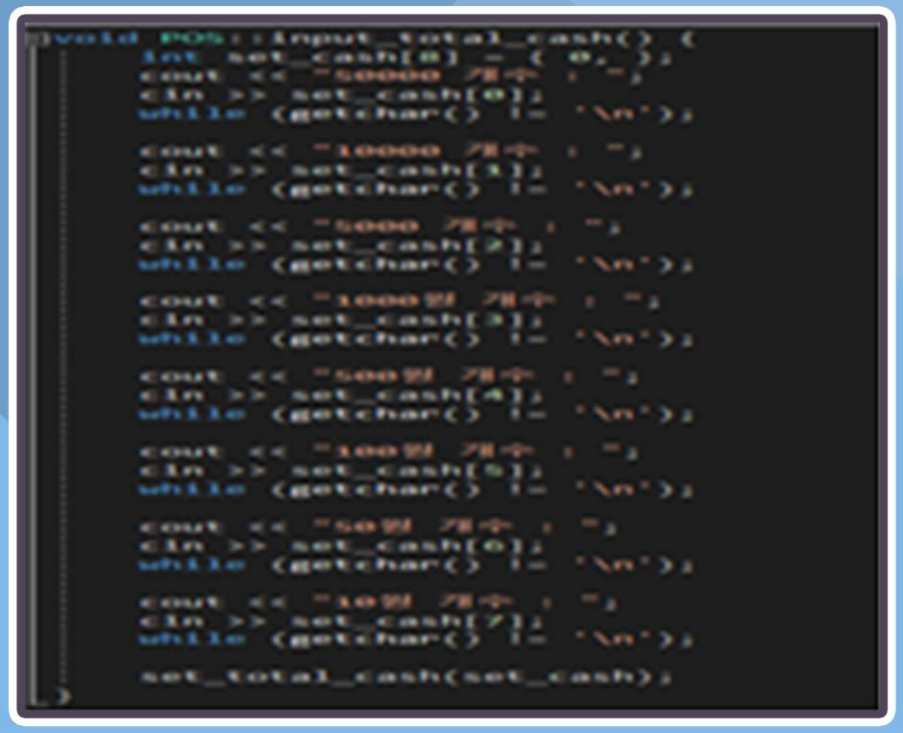
: A method for setting money in device.

```
void POS::productWrite(const char* filename) {
    ofstream outFile(filename);

    for (int i = 0; i < product.size(); i++) {
        outFile << product.at(i).getname() << "/" << product.at(i).getprice() << "/";
        for (int j = 0; j < product.at(i).getstock().size(); j++) {
            outFile << product.at(i).getvalid_date().at(j) << "." << product.at(i).getstock().at(j);
            if (j == product.at(i).getstock().size() - 1)
                outFile << "##/";
            else
                outFile << ".";
        }
        outFile << product.at(i).getdiscount() << "/" << product.at(i).getsell() << endl;
    }
    outFile.close();

}
```

3-4. POS

void input_total_cash()

: A method for writting money in device.



```
void POS::input_total_cash() {
    int set_cash[8] = { 0, };
    cout << "50000 개수 : ";
    cin >> set_cash[0];
    while (getchar() != '\n');

    cout << "10000 개수 : ";
    cin >> set_cash[1];
    while (getchar() != '\n');

    cout << "5000 개수 : ";
    cin >> set_cash[2];
    while (getchar() != '\n');

    cout << "1000원 개수 : ";
    cin >> set_cash[3];
    while (getchar() != '\n');

    cout << "500원 개수 : ";
    cin >> set_cash[4];
    while (getchar() != '\n');

    cout << "100원 개수 : ";
    cin >> set_cash[5];
    while (getchar() != '\n');

    cout << "50원 개수 : ";
    cin >> set_cash[6];
    while (getchar() != '\n');

    cout << "10원 개수 : ";
    cin >> set_cash[7];
    while (getchar() != '\n');

    set_total_cash(set_cash);
}
```

3-4. POS

void modify_product()

: A method that modifies products.

```
void POS::modify_cash() {
    int input;
    cout << "현금 입금 : 0 출금 : 1" << endl;
    cin >> input;
    while (getchar() != '\n');

    system("cls");

    if (input == 0) {
        add_total_cash();
    }
    else if (input == 1) {
        sub_total_cash();
    }
}
```

3-4. POS

void add_stock()

: A method that adds product's stock.

```
void POS::add_stock() {
    int index, num, date;
    while (1) {
        cout << "재고를 추가할 상품의 번호를 입력하시오 : ";
        cin >> index;
        while (getchar() != '\n');
        cout << endl;
        if (index >= 1 && index <= product.size())
            break;
    }
    while (1) {
        cout << "추가할 개수를 입력하시오 : ";
        cin >> num;
        while (getchar() != '\n');
        cout << endl;
        if (num >= 1)
            break;
    }

    calendar.reset_curr_time();
    vector<int> time = calendar.returnTime();

    while (1) {
        cout << "상품들의 유통기한을 입력하시오 (예 : 20191225) : ";
        cin >> date;
        while (getchar() != '\n');
        cout << endl;
        if (date >= 100000000) {
            cout << "잘못된 값이 입력되었습니다.";
            continue;
        }
        else if (date >= 10000 * time[0] + 100 * time[1] + time[2])   // 오늘 이후 날짜
            break;
        else {
            cout << "유통기한이 이미 지난 상품입니다.";
            return;
        }
    }

    product.at(index).addstock(num, date);
}
```

3-4. POS

void sub_stock()

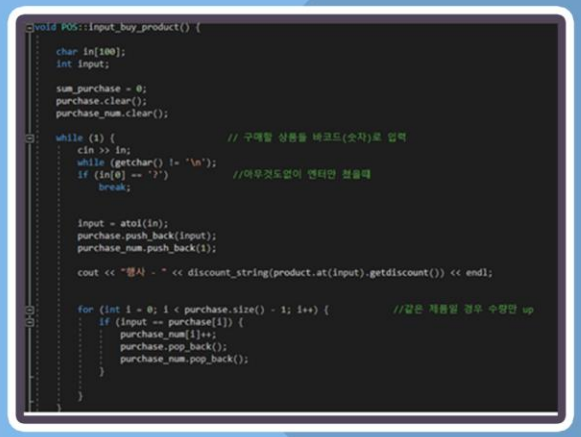
: A method for delete product which is not long before the expiration date.

```
void POS::add_stock() {
    int index, num, date;
    while (1) {
        cout << "재고를 추가할 상품의 번호를 입력하시오 : ";
        cin >> index;
        while (getchar() != '\n');
        cout << endl;
        if (index >= 1 && index <= product.size())
            break;
    }
    while (1) {
        cout << "추가할 개수를 입력하시오 : ";
        cin >> num;
        while (getchar() != '\n');
        cout << endl;
        if (num >= 1)
            break;
    }

    calendar.reset_curr_time();
    vector<int> time = calendar.returnTime();

    while (1) {
        cout << "상품들의 유통기한을 입력하시오 (예 : 20191225) : ";
        cin >> date;
        while (getchar() != '\n');
        cout << endl;
        if (date >= 100000000) {
            cout << "잘못된 값이 입력되었습니다.";
            continue;
        }
        else if (date >= 10000 * time[0] + 100 * time[1] + time[2])   // 오늘 이후 날짜
            break;
        else {
            cout << "유통기한이 이미 지난 상품입니다.";
            return;
        }
    }

    product.at(index).addstock(num, date);
}
```

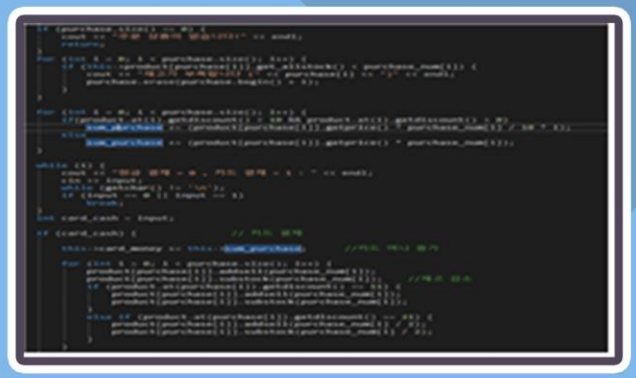
3-4. POS

void input_buy_product()

: A method that implements payment process.



```
void POS::Input_buy_product() {

    char in[100];
    int input;

    sum_purchase = 0;
    purchase.clear();
    purchase_num.clear();

    while (1) {                     // 구매할 상품들 바코드(숫자)로 입력
        cin >> in;
        while (getchar() != '\n');
        if (in[0] == '?')           //아무것도없이 엔터만 쳤을때
            break;


        input = atoi(in);
        purchase.push_back(input);
        purchase_num.push_back(1);

        cout << "행사 - " << discount_string(product.at(input).getdiscount()) << endl;


        for (int i = 0; i < purchase.size() - 1; i++) {          //같은 제품일 경우 수량만 up
            if (input == purchase[i]) {
                purchase_num[i]++;
                purchase.pop_back();
                purchase_num.pop_back();
            }

        }
    }
```

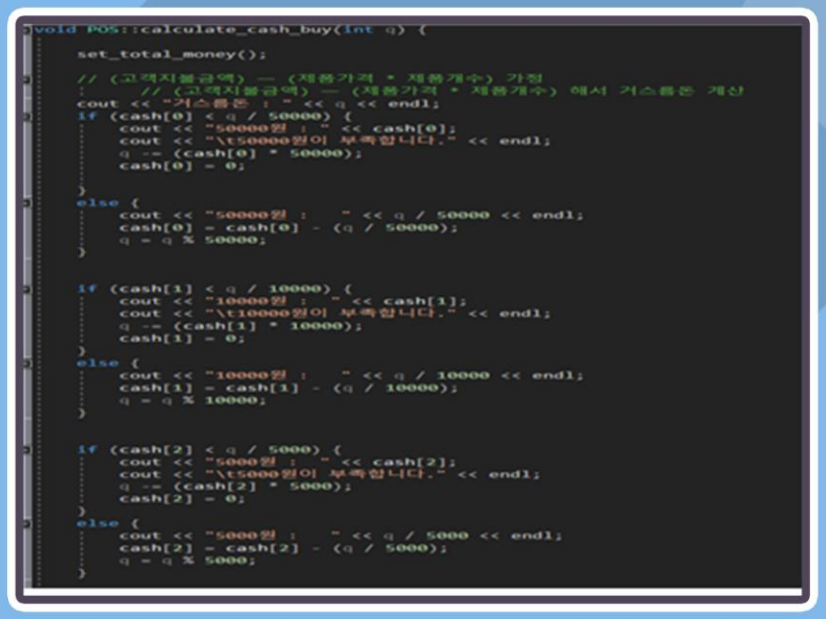


```
else {                  // 현금 결제
    while (1) {
        cout << "받은 금액 입력 : " << endl;
        cin >> this->get_money;
        while (getchar() != '\n');
        //qt에서 추가 현금 얼마짜리로 몇장 받는지
        if (this->get_money >= sum_purchase) {

            break;
        }
        cout << "금액이 부족합니다" << endl;
    }
    this->change = this->get_money - sum_purchase;          //거스름돈
    this->calculate_cash_buy(this->change);                 //현금 별 거스름돈 계산?
    for (int i = 0; i < purchase.size(); i++) {
        product[purchase[i]].addsell(purchase_num[i]);
        product[purchase[i]].substock(purchase_num[i]);         //재고 감소
        if (product.at(purchase[i]).getdiscount() == 11) {
            product[purchase[i]].addsell(purchase_num[i]);
            product[purchase[i]].substock(purchase_num[i]);
        }
        else if (product.at(purchase[i]).getdiscount() == 21) {
            product[purchase[i]].addsell(purchase_num[i] / 2);
            product[purchase[i]].substock(purchase_num[i] / 2);
        }
    }
}
Sleep(5000);
//영수증 - purchase 이용해서!!
}
```

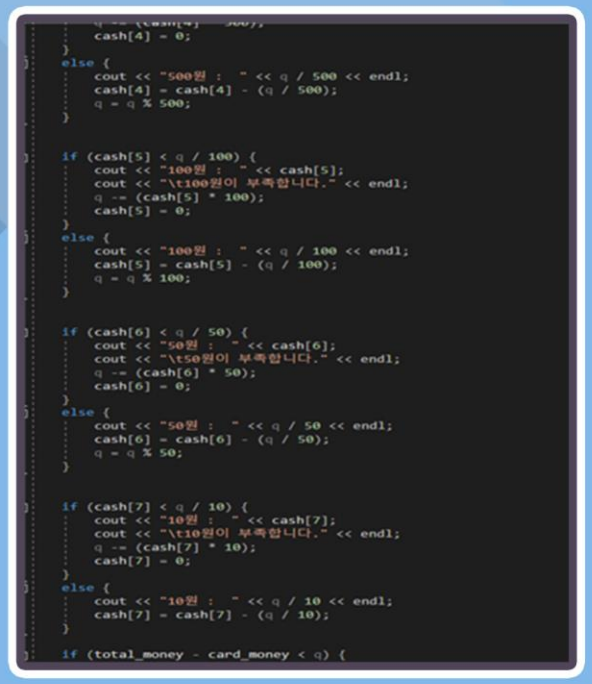
3-4. POS

void calculate_cash_buy(int q)

: A method that calculates exchange after payment.



```
void POS::calculate_cash_buy(int q) {

    set_total_money();

    // (고객지불금액) — (제품가격 * 제품개수) 가정
        // (고객지불금액) — (제품가격 * 제품개수) 해서 거스름돈 계산
    cout << "거스름돈 : " << q << endl;
    if (cash[0] < q / 50000) {
        cout << "50000원 : " << cash[0];
        cout << "\t50000원이 부족합니다." << endl;
        q -= (cash[0] * 50000);
        cash[0] = 0;

    }
    else {
        cout << "50000원 :   " << q / 50000 << endl;
        cash[0] = cash[0] - (q / 50000);
        q = q % 50000;
    }


    if (cash[1] < q / 10000) {
        cout << "10000원 : " << cash[1];
        cout << "\t10000원이 부족합니다." << endl;
        q -= (cash[1] * 10000);
        cash[1] = 0;
    }
    else {
        cout << "10000원 :   " << q / 10000 << endl;
        cash[1] = cash[1] - (q / 10000);
        q = q % 10000;
    }


    if (cash[2] < q / 5000) {
        cout << "5000원 : " << cash[2];
        cout << "\t5000원이 부족합니다." << endl;
        q -= (cash[2] * 5000);
        cash[2] = 0;
    }
    else {
        cout << "5000원 :   " << q / 5000 << endl;
        cash[2] = cash[2] - (q / 5000);
        q = q % 5000;
    }
```



```
        cash[4] = 0;
    }
    else {
        cout << "500원 :  " << q / 500 << endl;
        cash[4] = cash[4] - (q / 500);
        q = q % 500;
    }


    if (cash[5] < q / 100) {
        cout << "100원 : " << cash[5];
        cout << "\t100원이 부족합니다." << endl;
        q -= (cash[5] * 100);
        cash[5] = 0;
    }
    else {
        cout << "100원 :  " << q / 100 << endl;
        cash[5] = cash[5] - (q / 100);
        q = q % 100;
    }


    if (cash[6] < q / 50) {
        cout << "50원 : " << cash[6];
        cout << "\t50원이 부족합니다." << endl;
        q -= (cash[6] * 50);
        cash[6] = 0;
    }
    else {
        cout << "50원 :  " << q / 50 << endl;
        cash[6] = cash[6] - (q / 50);
        q = q % 50;
    }


    if (cash[7] < q / 10) {
        cout << "10원 : " << cash[7];
        cout << "\t10원이 부족합니다." << endl;
        q -= (cash[7] * 10);
        cash[7] = 0;
    }
    else {
        cout << "10원 : " << q / 10 << endl;
        cash[7] = cash[7] - (q / 10);
    }

    if (total_money - card_money < q) {
```

3-4. POS

void discount_string(int dis)

: A method that sets product's events.

```
string POS::discount_string(int dis) {
    switch (dis) {
    case 0:
        return "행사 없음";
    case 1:
        return "10% 할인";
    case 2:
        return "20% 할인";
    case 3:
        return "30% 할인";
    case 4:
        return "40% 할인";
    case 5:
        return "50% 할인";
    case 6:
        return "60% 할인";
    case 7:
        return "70% 할인";
    case 8:
        return "80% 할인";
    case 9:
        return "90% 할인";
    case 11:
        return "1+1 행사";
    case 21:
        return "2+1 행사";

    }
}
```

3-4. POS

void trash_product_clear()

: A method for disposes products that are out of date.

```
void POS::trash_product_clear() {
    int index, date;

    calendar.reset_curr_time();
    vector<int> time = calendar.returnTime();


    cout << "***유통기한 지난 제품 목록***" << endl;
    int num = 0;
    for (int i = 0; i < product.size(); i++) {
        for (int j = 0; j < product.at(i).getstock().size(); j++) {

            if (product.at(i).getvalid_date().at(j) < 10000 * time[0] + 100 * time[1] + time[2])
                continue;
            else {
                cout << i + 1 << ". " << product.at(i).getname() << endl;
                cout << product.at(i).getstock().at(j) << "개 - ";
                cout << product.at(i).getvalid_date().at(j) << "까지" << endl;
                num++;
            }


        }
        product.at(i).substock(num);
    }


    Sleep(5000);
    cout << endl << "제거 완료";
    Sleep(2000);
    productWrite("Product.txt");
}
```

3-5. Transaction

void add_transaction(vector<string> name, vector<int> price, vector<int> count, vector<int> discount, int sum_price)
: A method that adding transaction details.

void transactionRead(string filename)
: A method for reading text files with transaction details.

void print_receipt(int intdex)
: A method for printing receipt.

3-5. Transaction

void add_transaction(vector<string> name, vector<int> price, vector<int> count, vector<int> discount, int sum_price)

: A method that adding transaction details.

```
void Transaction::add_transaction(vector<string> name, vector<int> price, vector<int> count, vector<int>
    Tran input_t;

    input_t.number = transact.size() + 1;
    for (int i = 0; i < name.size(); i++) {
        input_t.name.push_back(name.at(i));
        input_t.price.push_back(price.at(i));
        input_t.count.push_back(count.at(i));
        input_t.discount.push_back(discount.at(i));
    }
    input_t.sum_price = sum_price;


    transact.push_back(input_t);

}
```

3-5. Transaction

void transactionRead(string filename)

: A method for reading text files with transaction details.

```
void Transaction::transactionRead(string filename) {

    ifstream inFile(filename);

    while (!inFile.eof()) {

        Tran input_t;

        char inputLine[200] = { 0, };
        inFile.getline(inputLine, 200);

        if (inputLine[0] == NULL) { break; }
        vector<string> r_name;
        vector<int> r_price;
        vector<int> r_count;
        vector<int> r_discount;
        int r_number, r_sum_price;

        string r_str;

        input_t.number = atoi(strtok(inputLine, "/"));
        char* r_product_index = strtok(NULL, "/");
        input_t.sum_price = atoi(strtok(NULL, "/"));


        int r_index;
        char* r_c;
```

```
    while (1) {
        if (c) {
            r_c = strtok(r_product_index, ".#");
            c = false;
        }
        else {
            r_c = strtok(NULL, ".#");
            if (r_c == NULL)
                break;
            r_index = atoi(r_c);
        }

        input_t.name.push_back(r_c);

        r_index = atoi(strtok(NULL, ".#"));
        input_t.price.push_back(r_index);

        r_index = atoi(strtok(NULL, ".#"));
        input_t.count.push_back(r_index);

        r_index = atoi(strtok(NULL, ".#"));
        input_t.discount.push_back(r_index);
    }


    transact.push_back(input_t);

}
```

3-5. Transaction

void print_receipt(int intdex)

: A method for printing receipt.

```
void Transaction::print_receipt(int index) {        // 영수증 출력

    cout << " **********************영수증************************" << endl;
    cout << "대표자 이름 : " << rep_name << endl;
    cout << "가게 이름 : " << store_name << endl;
    cout << "전화번호 : " << phone_num << endl;
    cout << "가게 주소 : " << address << endl;
    cout << "와이파이 비밀번호 : emart3474 " << endl;
    cout << "***\t 품명 \t 단가 \t 수량 \t 행사 \t\t 금액 ***" << endl;
    for (int i = 0; i < transact.at(index).name.size(); i++) {
        cout << i + 1 << ".\t" << transact.at(index).name.at(i) << "\t" << transact.at(index).price.at(i
        cout << discount_string(transact.at(index).discount.at(i)) << "\t" << transact.at(index).price.a

    }
    cout << "총 금액 : " << transact.at(index).sum_price << endl;
    cout << " ****************************************************" << endl;
    ////////////////////////////////////////////////////////////////////////////////////////////////////

}
```

3-6. Customer

bool check_name(string _name)

: A method that checking whether customers name is same with parameter.

bool check_tel(string _tel)

: A method that checking whether customers tel number is same with parameter.

3-6. Customer

bool check_name(string _name)

: A method that checking whether customers name is same with parameter.

```
bool Customer::Check_name(char* _name) {
    if (name == _name) return true;
    else return false;
}
```

3-6. Customer

bool check_tel(string _tel)

: A method that checking whether customers tel number is same with parameter.

```
bool Customer::Check_tel(char* _tel) {
    int count = 0;
    for (int index = 0; index < 4; index++)
        if (tel[index] == _tel[index]) count++;
    if (count == 4) return true;
    else return false;
}
```

Features 

- We can manage customers by analyzing customers' age, sex etc.
- We can save and use points.
- We can manage product's stock and discard
- There are event products ex) 10% discount.

Features 

- We can decide staff's pay by checking negligence using staff's start hour and end hour.

- We can give wages using tax, extra pay, non-duty allowance.

- We can add and delete staff.

- We can manage staff's statement.

Features

- We can manage events / holidays / staff's schedule in calendar
- We can save transaction lists in MMA and print receipt.
- We can decide cash / card pay and calculate exchange
- We can manage cash in MMA.

Features 

- We can save and manage product's sales volume.
- We can add and delete products.

Thank You!!