**2022년 현대모비스 프로젝트 6조**

| **[ 자동차 주차 관리 시스템]** |
| --- |
| **소프트웨어 설계서** |

2022년 02월 26일

문서번호 : 2022-Doc-001

팀 명 : 6조

팀 원 : 정해경, 안현빈, 권현지

# 제/개정 이력

| **버전** | **날짜** | **작성자 성명** | **제/개정사항** | **비 고** |
| --- | --- | --- | --- | --- |
| v0.1 | 22/02/26 | 팀원 모두 | sw설계서 초안 작성 |  |
| v0.2 | 22/02/27 | 팀원 모두 | 동적 다이어그램 추가 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목차**

[**제/개정 이력**](#_xm5d692fzoyy) **2**

[**1. 서론**](#_nbzh23iid8bx) **4**

[**2. 소프트웨어 아키텍처**](#_ffflwri4k067) **4**

[**3. 클래스 설계**](#_hnfyp0fcci4n)**(자료구조) 10**

[3.1.1 클래스 설명 및 설계](#_wvziqat18wo6) 10

[(1) C-001 : Parking](#_sc28siw6af0q) 10

[3.1.2 멤버함수(메소드) 설계](#_14vr22sq9rhw) 11

[(1) M-001 : enterCar()](#_2dx5tdar4904) 11

[(2) M-002 : exitCar()](#_yzuzkbhs9nu8) 13

[(3) M-003 : checkEmptyArea()](#_t74o7tphhdv8) 14

[(4) M-004 : showPos()](#_8mo0ll4x5sal) 14

[3.2.1 클래스 설명 및 설계](#_4e66m9vwrp11) 15

[(2) C-002 :Mem](#_h19h1y8y851c) 15

[3.2.2 멤버함수(메소드) 설계](#_v9nh54900t8f) 16

[(1) M-001 : findPhoneNum()](#_yc5u8uyk1lmi) 16

[(2) M-002 : isMem()](#_lzx6dxyhzq) 17

[(3) M-003 : registerMem()](#_6i10jdcdnyad) 17

[3.3.1 클래스 설명 및 설계](#_qc1jz0cs598n) 18

[(3) C-003 : File](#_we071je26mnt) 18

[3.3.2 멤버함수(메소드) 설계](#_pl61wfkhkp07) 18

[(1) M-001 : saveExit()](#_wl9br1yvpzep) 19

[(2) M-002 : savePos()](#_noa35xty79dv) 19

[(3) M-003 : loadPos()](#_kw6i9e1on1ei) 20

[(4) M-004 : loadMem()](#_629khoqi9ete) 21

[3.4.1 클래스 설명 및 설계](#_x51uceu8aotq) 22

[(4) C-004 :Car](#_spobapl1paki) 22

[3.4.2 멤버함수(메소드) 설계](#_fws5e75mp1ml) 22

[(1) M-001 : calc\_cost()](#_nl2h5vrf2hc5) 22

[(2) M-002 : calc\_time()](#_ab8p3zrvm3ow) 23

[(3) M-003 : find\_pos()](#_b4ys66y7xh3d) 23

[3.5.1 클래스 설명 및 설계](#_bhw5qgoxz7r) 24

[(5) C-005: Admin](#_b104s81f28o8) 24

[3.5.2 멤버함수(메소드) 설계](#_t94dcjpjnwj3) 25

[(1) M-001 : loadExit()](#_fayqevaxnpa5) 25

[(2) M-002 : perDay()](#_hpk6ilaq1c5u) 25

[(3) M-003 : perMonth()](#_ly39pzzfw9cl) 27

[**4. 인터페이스 설계**](#_4xzf6ofxsarq) **28**

[**5. 구현 기술 설계**](#_hxu3ksy9a8m2) **29**

[**6. 세부일정**](#_8f4hcdp3oh51) **30**

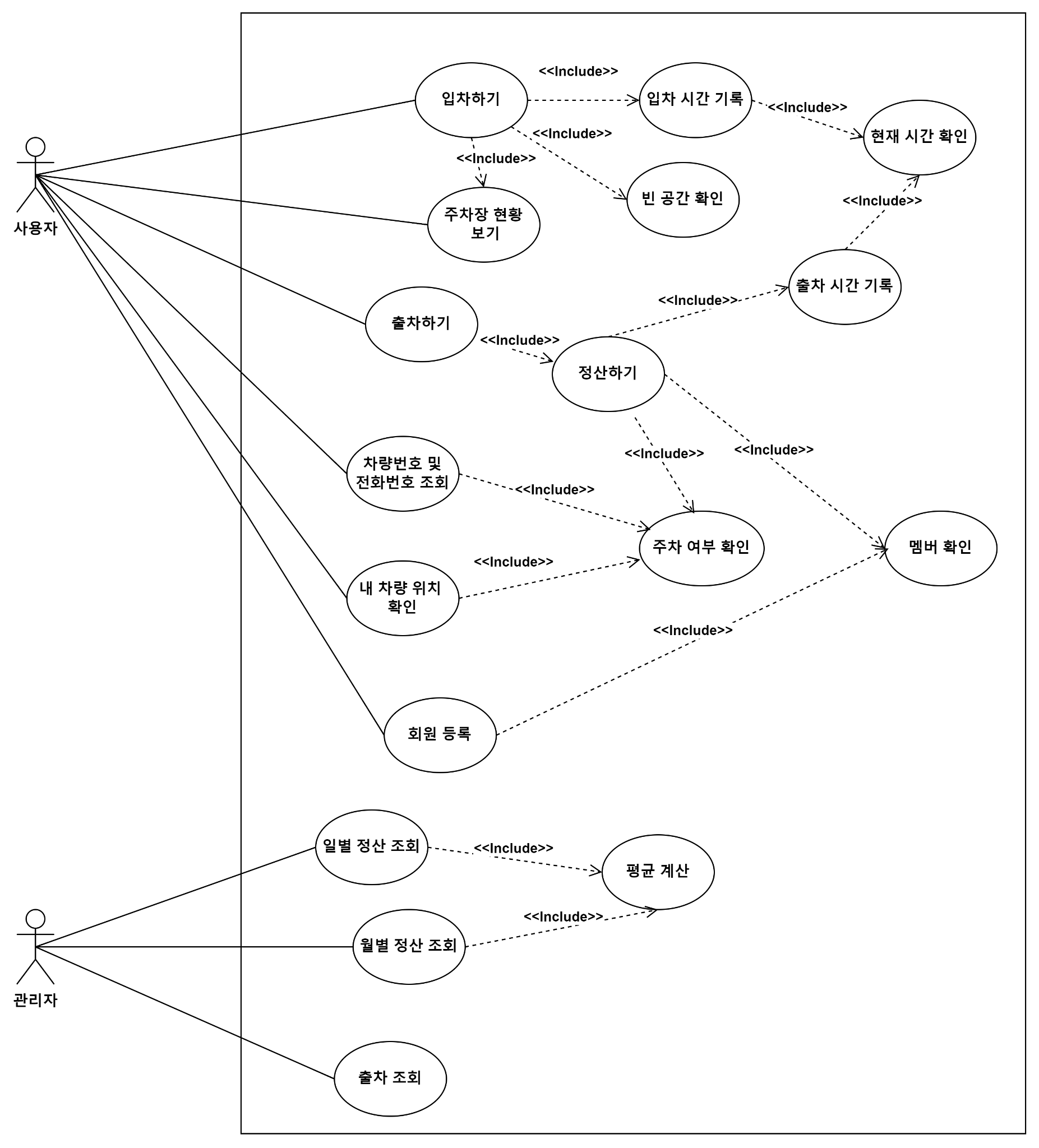
# **1. 서론**

**1.1 목적 및 범위**

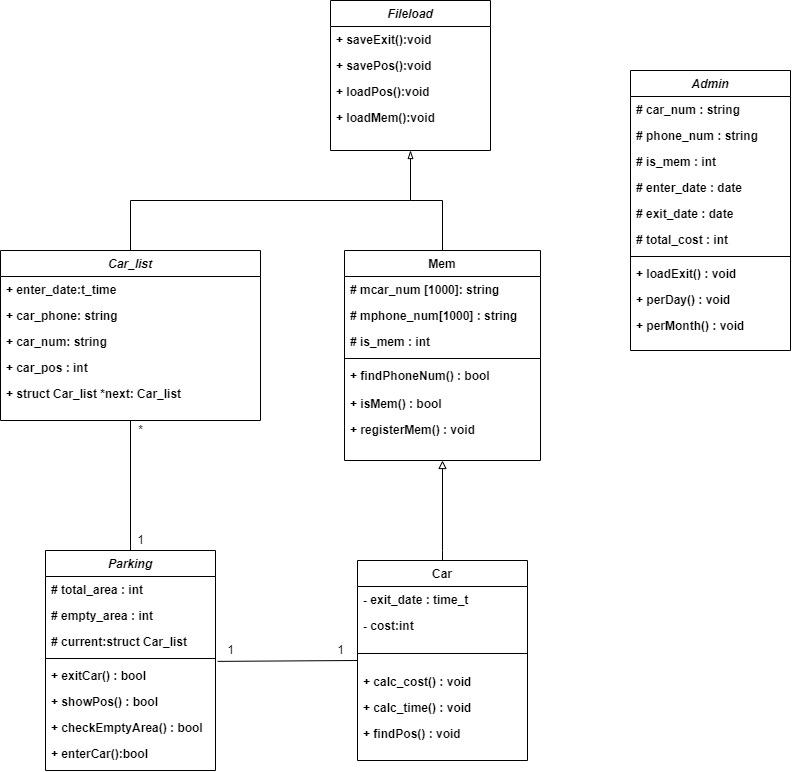
본 소프트웨어 설계서는 원활한 주차 관리를 위해 요구되는 각 기능의 설계에 대해 분석하여 어떻게 동작하는지, 어떤 동작을 하는지 명확하게 설계하는 것을 목적으로 한다. 즉, 요구사항 분석 결과를 활용하여 아키텍처 및 모듈별 상세 설계 결과를 문서화 한다

# **2. 소프트웨어 아키텍처**

**2.1 유스케이스 다이어그램**



**2.2 클래스 다이어그램**



**2.3 시퀀스 다이어그램**

| **<입차>** |
| --- |
|  |
| **<출차>** |
|  |
| **<차량번호 및 전화번호 조회>** |
|  |
| **<내차량 위치 확인>** |
|  |
| **<회원등록>** |
|  |
| **<일별 정산 조회>** |
|  |
| **<월별 정산 조회>** |
|  |

| 클래스명 | 기능 설명 | 담당자 |
| --- | --- | --- |
| **Parking** | 주차장에 관련된 현황 파악과 입차 및 출차를 위한 클래스 | 정해경 |
| **Mem** | 회원 등록과 조회에 관련된 기능을 위한 클래스 | 정해경 |
| **File** | 사용자와 관리자가 요청한 주문을 확인하고 관리하기 위한 주체들을 포함하는 클래스 | 권현지 |
| **Car** | 사용자 입차/출차 시간을 통한 정산관리 클래스 | 권현지 |
| **Admin** | 관리자가 정산과 회원과 관련된 기능을 관리하기 위한 클래스 | 안현빈 |

# **3. 클래스 설계**

## **3.1.1 클래스 설명 및 설계**

### (1) C-001 : Parking

| 설 명 | 주차장에 관련된 현황 파악과 입차 및 출차를 위한 클래스 | | |
| --- | --- | --- | --- |
| 멤버 변수 | # total\_area : int  # empty\_area : int  # current: struct Car\_list | | |
| 멤버 함수 | + enterCar() : bool  + checkEmptyArea() : bool  + exitCar() : bool  + printPos() : bool | | |

## **3.1.2 멤버함수(메소드) 설계**

### (1) M-001 : enterCar()

| 파라메터 | current \*\*HEAD,current \*\*TAIL |
| --- | --- |
| 세부 처리 로직 | |
| public bool enterCar(current \*\*HEAD,current \*\*TAIL) {  if Head null 확인  **//1current 통해 showPos//현재 주차장 현황**  struct current \*tmp;  tmp=(struct current\*)malloc(sizeof(struct current))  if null확인  showPos(HEAD);  **//2. 선택->checkEmptyArea()->selectArea()**  cin>>tmp->car\_pos;  if(checkEmptyArea(tmp->car\_pos)){  cin >> tmp->car\_num >>tmp-> car\_phone; }  else{  cout<< “주차되어있는 공간입니다.\n”;    **//3. 입차 시간 기록 car \*p p->enter\_date=calc\_time()**  Car \*ttime;  ttime.calc\_time(&tmp->enter\_date)현재시간  tmp->next = NULL;  \*TAIL->next = tmp;  \*TAIL=tmp; | |

### (2) M-002 : exitCar()

| 파라메터 | current \*\*HEAD, current \*\*TAIL |
| --- | --- |
| 세부 처리 로직 | |
| public bool exitCar(current \*\*HEAD,current \*\*TAIL) {  //1.Car:printPos->Parkinglot->p\_position배열을 읽고 있으면->Car:car\_pos  없으면->”없다고 출력”  cin >> car\_num; //출차할 차 번호 입력  struct current \*tmp,\*before;  Car \*p;  tmp = before = \*HEAD  while(tmp){  if(tmp-> car\_num == car\_num){  p->calc\_time(&p->exit\_date) // 현재 시간을 받아오고  p->calc\_cost(tmp, &p); //정산을 하고  saveExit(tmp, p->cost,p->exit\_date);  if(tmp ==(\*HEAD)) {// 노드 위치 케이스 확인 필요!!!!!!!!!!(헤드, 테일, 중간)  (\*HEAD)=(\*HEAD)->next;  }else if(tmp == (\*TAIL)){  (\*TAIL) = before;  (\*TAIL)->next = NULL;  }else {  before->next = tmp ->next;  }  free(tmp); break;  before = tmp;  tmp = tmp -> next; | |

### (3) M-003 : checkEmptyArea()

| 파라메터 | current \*HEAD, string car\_pos |
| --- | --- |
| 세부 처리 로직 | |
| struct current \*tmp;  tmp=HEAD  while(tmp){  if(tmp->car\_pos == car\_pos){  return true;  }  tmp = tmp -> next;  }  return false; | |

### (4) M-004 : showPos()

| 파라메터 | current \*HEAD, int car\_pos |
| --- | --- |
| 세부 처리 로직 | |
| int p\_arr[20];  struct current \*tmp;  tmp = HEAD;  while(tmp)(  p\_arr[tmp->car\_pos] = 1;  tmp = tmp->next;  }  //p\_arr를 이쁘게 프린트  for(int i=0;i<5;li++){  for(int j =0;j<4;j++){  if(p\_arrr[i]==1)  cout<<’X’;  else  cout<< (i)\*5+(j+1) ; | |

## **3.2.1 클래스 설명 및 설계**

### (2) C-002 :Mem

| 설 명 | 회원 등록과 조회에 관련된 기능을 위한 클래스 | | |
| --- | --- | --- | --- |
| 멤버 변수 | # mcar\_num[1000] : string  # mphone\_num[1000] : string  # is\_mem : int | | |
| 멤버 함수 | + findPhoneNum() : bool  + registerMem() : void  + isMem() : bool | | |

## **3.2.2 멤버함수(메소드) 설계**

### (1) M-001 : findPhoneNum()

| 파라메터 | current \*HEAD |
| --- | --- |
| 세부 처리 로직 | |
| struct current \*tmp;  tmp=HEAD  string car\_num;  cin>>car\_num;  while(tmp){  if(tmp->car\_num == car\_num){  cout<< tmp->car\_phone;  return true;  }  }  return false; | |

### (2) M-002 : isMem()

| 파라메터 | string car\_num |
| --- | --- |
| 세부 처리 로직 | |
| map<string, string> \*mem\_map;  Fileload \*fp;  fp->loadMem(mem\_map); // 멤버 받아오기  while(true){  if(mem\_map->find()==mem\_map.end()){  return false;  }  else if((key == car\_num){  return true;  }  else{  return false;  } | |

### (3) M-003 : registerMem()

| 파라메터 |  |
| --- | --- |
| 세부 처리 로직 | |
| string car\_num, string phone\_num;  cin >> car\_num >> phone\_num;  ofstream fout;  fout.open(“멤버.txt”,ios::out |ios::app);  fout<<car\_num<<’\t’<<phone\_num<<endl;  fout.close();  return; | |

## **3.3.1 클래스 설명 및 설계**

### (3) C-003 : File

| 설 명 | 사용자와 관리자가 요청한 주문을 확인하고 관리하기 위한 주체들을 포함하는 클래스 | | |
| --- | --- | --- | --- |
| 멤버 변수 | x | | |
| 멤버 함수 | + saveExit() : void  + savePos() : void  + loadPos() : void  + loadMem() : void | | |

## **3.3.2 멤버함수(메소드) 설계**

### (1) M-001 : saveExit()

| 파라메터 | current \*\*tmp, int cost, time\_t exit\_date |
| --- | --- |
| 세부 처리 로직 | |
| File \*fp = fopen(“경로\total.txt”, “a”);  char \*arr[]={tmp->car\_num,tmp->car\_phone,tmp->enter\_date,exit\_date,cost};  fwrite(arr, sizeof(int), 1, fp);  fclose(fp); | |

### (2) M-002 : savePos()

| 파라메터 | current \*HEAD |
| --- | --- |
| 세부 처리 로직 | |
| File \*fp = fopen(“경로\주차.txt”, “w”);  struct current \*tmp;  tmp = HEAD;  while(tmp){  char arr[]={tmp->car\_num,tmp->car\_phone,tmp->enter\_date,tmp->car\_pos};  fwrite(arr, sizeof(current), 1, fp);  tmp = tmp -> next;  }  fclose(fp); | |

### (3) M-003 : loadPos()

| 파라메터 | current \*\*HEAD,current \*\*TAIL |
| --- | --- |
| 세부 처리 로직 | |
| File \*fp = fopen(“경로\주차.txt”, “r”);  current \*temp;  char\* cur  strcat  while(1){  temp=(struct current\*) malloc(sizeof(struct current));  if(temp==NULL)확인 {//예외처리  }  if(fscanf(fp, “%s %s %s %s %d”, temp->car\_num, temp->car\_phone,temp->enter\_date, temp->car\_pos)!=EOF) {  if(\*HEAD == NULL){  \*HEAD =\*TAIL= temp;  }  else{  (\*TAIL)->next=temp;  \*TAIL=temp;  }  }    fclose(fp); | |

### (4) M-004 : loadMem()

| 파라메터 | Map <string, string> \*\*mem\_map; |
| --- | --- |
| 세부 처리 로직 | |
| ofstream fout(“경로\회원.txt”);  Map <string, string> mem\_map;  Mem \*m;  char c;  string tmp;  while(1){  string car\_num;  string phone\_num;  if(c=fout.get()!=EOF){  if(c=’ ‘){  car\_num =tmp;  tmp=””;  else if(c==’\n’){  phone\_num = tmp;  tmp=””;  }  else{  tmp+=c;  }    mem\_map.insert(make\_pair(car\_num, phone\_num));  }  } | |

## 

## **3.4.1 클래스 설명 및 설계**

### (4) C-004 :Car

| 설 명 | 사용자 입차/출차 시간을 통한 정산관리 클래스 | | |
| --- | --- | --- | --- |
| 멤버 변수 | #exit\_date:time+t  #cost:int | | |
| 멤버 함수 | + calc\_cost() : void  + calc\_time() : void  + findPos() : void | | |

## **3.4.2 멤버함수(메소드) 설계**

### (1) M-001 : calc\_cost()

| 파라메터 | struct current tmp, Car \* p |
| --- | --- |
| 세부 처리 로직 | |
| if( is\_mem(tmp->car\_num)){  calc\_time(&p->exit\_date)  p->cost=(p->exit\_date - tmp->enter\_date)) \* 1000 \*0.8 //20%할인  }  //할인 x  p->cost=(p->exit\_date - tmp->enter\_date)) \* 1000  return; | |

### (2) M-002 : calc\_time()

| 파라메터 | string \*date; |
| --- | --- |
| 세부 처리 로직 | |
| date = 현재시간;  return; | |

### (3) M-003 : find\_pos()

| 파라메터 | struct current \* HEAD |
| --- | --- |
| 세부 처리 로직 | |
| string car\_num;  cin >> car\_num;  struct current tmp;  tmp = HEAD;  while(tmp){  if(tmp->car\_num == car\_num){  cout << tmp->car\_pos;  return;  }  }  cout<<”찾으시는 차량이 주차되어 있지 않습니다.<<endl;  return; | |

## 

## **3.5.1 클래스 설명 및 설계**

### (5) C-005: Admin

| 설 명 | 관리자가 정산과 회원과 관련된 기능을 관리하기 위한 클래스 | | |
| --- | --- | --- | --- |
| 멤버 변수 | #car\_num : string  #phone\_num : string  #is\_mem : int  #enter\_date : time\_t  #exit\_date : time\_t  #total\_cost : int | | |
| 멤버 함수 | + loadExit() : void  + perDay(): void  + perMonth() : void | | |

## **3.5.2 멤버함수(메소드) 설계**

### (1) M-001 : loadExit()

| 파라메터 |  |
| --- | --- |
| 세부 처리 로직 | |
| //전역 vector<vector<string> > v;  fstream fin(“경로\total.txt”,ios::in)  vector<string> temp;  if (!fin)->예외처리e  while(1){  if(temp==NULL)확인 {//예외처리  }  fileread(temp,fin);//라인별로 읽기  v.push\_back(temp);  fin.close() | |

### (2) M-002 : perDay()

| 파라메터 |  |
| --- | --- |
| 세부 처리 로직 | |
| //차량번호,전화번호,입차일,출차일,정산요금  string temp= v[v.size()-1][3];  string firstday= v[v.size()-1][3];  for(int i = v.size()-1; i>=0; i—){//7일간  int sum=0;  if(firstday-temp==7){  break;  }  if(temp==v[i][3]){  sum+=v[i][4];  }  else{  cout<<temp<<”의 총 정산요금”<<sum;  sum=v[i][4];  temp=v[i][3];  }  temp= v[i][3]//일자까지만;  }  } | |

### (3) M-003 : perMonth()

| 파라메터 |  |
| --- | --- |
| 세부 처리 로직 | |
| //차량번호,전화번호,입차일,출차일,정산요금  string temp= v[v.size()-1][3];  string firstmonth= v[v.size()-1][3];  for(int i = v.size()-1; i>=0; i—){//12개월  int sum=0;  if(firstmonth-temp==12){  break;  }  if(temp==v[i][3]){//월까지만 같으면  sum+=v[i][4];  }  else{  cout<<temp<<”의 총 정산요금”<<sum;  sum=v[i][4];  temp=v[i][3];  }  temp= v[i][3]//월까지만;  }  } | |

# **4. 인터페이스 설계**

**4.1 사용자 인터페이스**

|  |  |
| --- | --- |
| 메인 화면 | 입차 |
|  |  |
| 출차 | 내 주차 이력 |
|  |  |
| 차주 조회 | 내 차량 위치 확인 |
|  |  |
|  |  |
| 관리자 메뉴 관리 | |

# **5. 구현 기술 설계**

| **구현 언어** | - c,c++ |
| --- | --- |
| **운영 체제** | - Raspbian(Linux) |
| **툴** | - Mobaxterm, Visual studio code |

# **6. 세부일정**

