2021-1 객체지향프로그래밍

2차 프로그래밍 과제

[주차장 관리 프로그램]

미디어학과 / 2학년

201821133 한규정

실습반번호 : 3

목차

1. 서론

1) 프로그램 설명

- Vehicle class의 subclass로 여러 종의 차량 class가 존재하며, ParkingCenter class에는 주차 혹은 출차한 차량을 저장하는 역할을 한다. Time class는 차량의 입차시간을 문자열로 받아 생성되고, 시간 차 계산 및 GregorianCalender를 이용하여 시간의 유효성을 검사한다.

특정 명령을 받아 입출차를 진행하고, 주차정보 혹은 수입을 출력할 수 있다.

2) 기능

1. 입차 : 차량번호, 차종, 입차시간, 선택사항을 입력받아 주차장에 해당 차량을 입차시킨다.
2. 출차 : 차량번호와 출차 시간을 입력받아 해당 차량을 주차장에서 출차시키고, 주차시간에 따른 주차요금을 계산하여 보여준다.
3. 주차차량 보기 : 주차번호와 시간을 입력받아 해당 차량의 정보를 보여준다. 가솔린, 벤의 경우 각각 배기량과 크기를 보여주고, 전기차의 경우 입력한 시간에 대한 충전량을 보여준다.

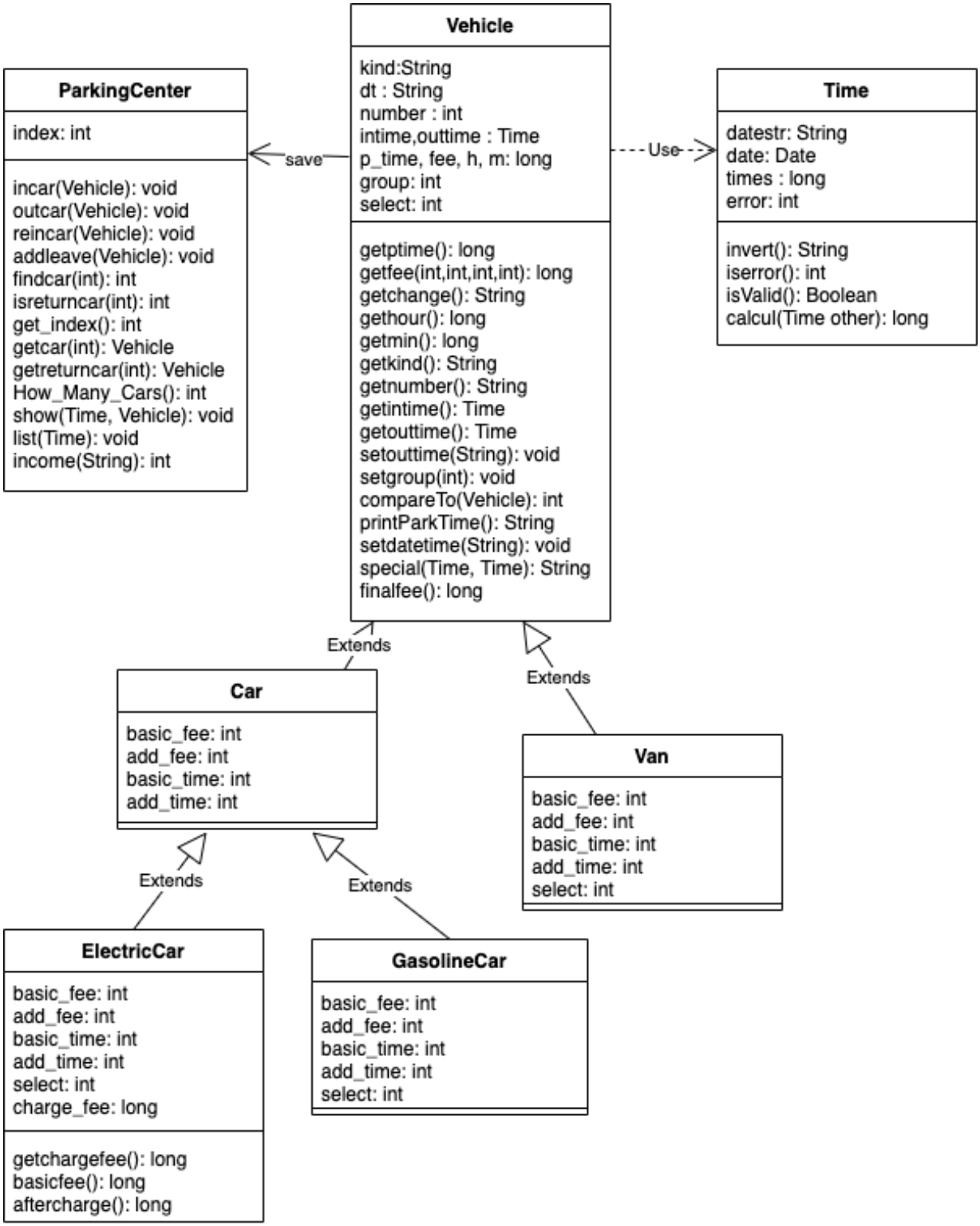
4. 주차차량 전체 리스트 보기 : 모든 주차차량을 종류별(가솔린 차량, 전기차, 벤 순서) 및 입차시간순으로 정렬하여 차량정보를 보여준다. 가솔린, 벤의 경우 각각 배기량과 크기를 보여주고, 전기차의 경우 입력한 시간에 대한 충전량을 보여준다.
5. 총 수입 보기 : 입력날짜의 총 수입을 출력한다.

3) 구현 완성도 표

분류	설명	완성도(%)
기능	입차하기	100
	출차하기(요금 계산)	100
	주차차량 정보 불러오기	100
	주차차량 리스트 출력하기(차종별 정렬하기)	100
	입력 날짜의 총 수입 출력하기	100
예외처리 / 유효성 검사	잘못된 형식의 날짜와 시간 입력 처리	100
	날짜 유효성 검사	100
	차량 번호 오류	100
	입차시 주차차량의 차량번호 입력	100
	입차시 출차차량과 같은 차량번호 입력(차종, 특이사항 검사)	100
	입차시 차종, 차종에 따라 선택사항 검사	100
	주차장 만석	100
	초기 주차용량 입력 값 검사	100
	명령의 시간 순서 검사, 입출차 시간 검사	100
	초기 주차용량 입력 값 검사	100

2. 분석 / 설계 (본론)

1) UML Class Diagram



2) Instance variable 설명

- **Vehicle class**

  Vehicle class는 abstract 클래스로서 Car class와 Van class를 상속한다. Kind(차종), number(차량번호), datetime(String형: 입력된 날짜,시간), select(특이사항)의 parameter를 받아서 생성할 수 있다. 생성할 때 문자형인 datetime을 통해 Time 객체를 만들고 그것이 입차 시간이 된다.

- **Car class**

  Car class는 super class인 Vehicle class에게 상속된다. 벤이 아닌 승용차로 분류되는 차량을 나타내며, 기본주차시간(basic_time)이 30으로 설정하여 생성한다..

- **GasolineCar class**

  GasolineCar class는 Car class에게 상속된다. 네 번째 정수형 매개변수를 받아서 생성할 때 배기량을 나타내는 displacement 변수에 대입한다. Displacement가 1000 미만이면 그렇지 않은 것의 기본요금(basic_fee)과 추가요금(add_fee)이 절반으로 설정되어 생성된다. 생성시 setgroup()에 의해 1번째 종으로 분류된다.

- **ElectricCar class**

  ElectricCar class는 Car class에게 상속된다. 네 번째 정수형 매개변수를 받아서 생성할 때 충전량을 나타내는 charge 변수에 대입한다. 기본요금(basic_fee)과 추가요금(add_fee)이 설정되어 생성된다. 생성시 setgroup()에 의해 2번째 종으로 분류된다.

- **Van class**

  Van class는 super class인 Vehicle class에게 상속된다. 네 번째 정수형 매개변수를 받아서 생성할 때 벤의 크기를 나타내는 size 변수에 대입한다. 기본주차시간(basic_time)과 기본요금(basic_fee)가 0이고 추가요금(add_fee)가 size에 따라 다르게 설정하여 생성된다. 생성시 setgroup()에 의해 3번재 종으로 분류된다.

- **ParkingCenter class**

  주차되어 있는 차량을 가져오기 위한 class이며 ArrayList를 통해 Vehicle 객체들을 저장하는 역할을 한다..

- **Time class**

  SimpleDateFormat을 이용하여, 특정 문자열로 입력된 날짜와 시간을 받고 입력이 옳다면 Time 객체로 생성한다. 입력이 틀리다면 error 변수에 값을 대입하여 geterror()로 보낸다.

3) 주요 Method 설명

- **Vehicle class**

Method	설명	Data
getkind()	차종을 반환함.	-
getnumber()	차량번호를 반환함.	-
getintime()	입차시간을 반환함.	-
getouttime()	출차시간을 반환함.	-
setouttime(String outtimestr)	출차시간을 설정함.	-
setgroup(int group)	그룹으로 설정함.	group : 그룹 숫자
getselect()	특이사항을 반환함.	-
gethour()	출차 – 입차를 하여 나온 주차 시간의 "시간"을 반환함.	-
getmin()	출차 – 입차를 하여 나온 주차 시간의 "분"을 반환함.	-
getptime()	출차시간 – 입차시간, 즉 주차 시간을 반환함.	-
getfee(int bf, int af, int bt, int at)	주차요금을 계산하는 메소드. 기본시간과 주차시간에 따라 if 문으로 나누어 추가요금과 기본 요금을 더한 주차요금을 반환함.	addfeetime : 추가요금으로 처리해야하는 시간 addfee: 추가요금
finalfee()	최종요금을 계산하는 abstract 메소드.	-
special(Time t1, Time t2)	list와 show 에서 특이사항 출력에 이용되는 abstract 메소드. 문자열을 반환	-
printParkTime()	leave 명령어에서 주차요금 등을 출력하는 메소드	-
getdatetime()	차량 입차시간(문자열) 반환	
setdatetime(String dt)	datetime(문자열 날짜시간)을 변경	-
getchange()	datetime 문자열의 시간 부분을 00으로 설정하여 income 명령에서 날짜만 같은 것들끼리 분류되도록 함. 바뀐 문자열을 반환함.	index newString: 변환되는 문자열
compareTo(Vehicle other)	list 명령에서 group별로 정렬하도록 하는 비교 메소드.	-

- **ElectricCar class**

Method	설명	Data
basicfee()	오버라이딩 / 주차요금을 반환함.	-
getchargefee()	완충에 필요한 시간과 주차시간에 따라 If문으로 나누어 충전시간을 계산하고 그에 따라 충전요금을 반환함.	t = (60 – 현재 충전량) * 2 / 완충에 필요한 시간
getcharge()	충전량을 반환함.	-
aftercharge()	충전 후 충전량을 반환함.	a : 기존 충전량 + 충전한 량

- **Van, Car, Gasoline class : 별도 메소드 없음.**

- **Time class**

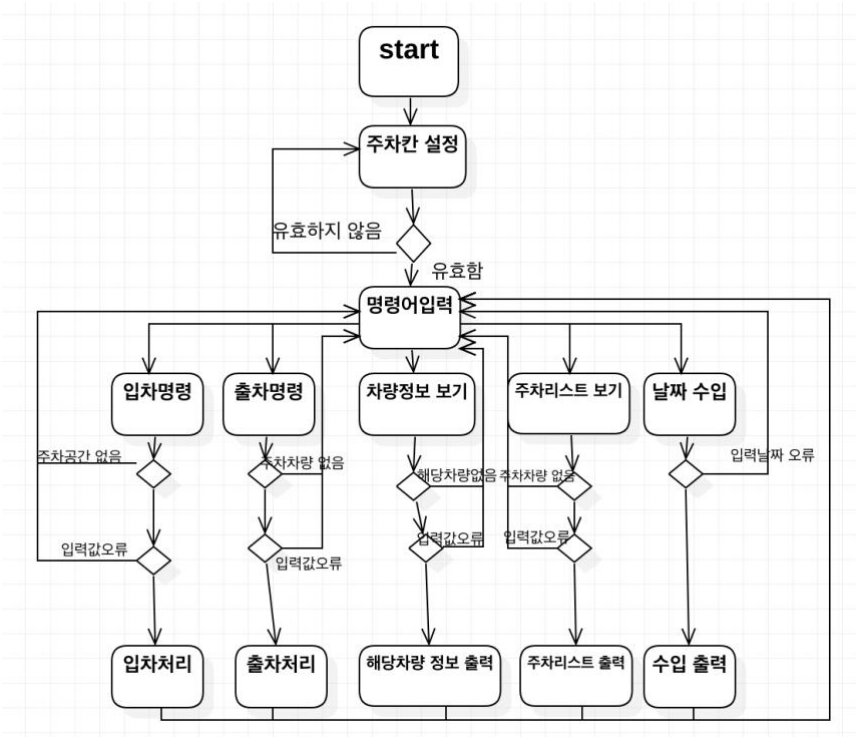
Method	설명	Data
invert()	yyyy MM dd hh mm 꼴의 문자열을 yyyy-MM-dd hh:mm 꼴로 변환시켜주는 메소드	-
iserror()	생성자할때 설정된 error를 반환하는 메소드	-
isValid()	GregorianCalendar를 이용해 윤년인지 검사하고, 날짜에 대한 유효성 검사를 하는 메소드	-
calcul(Time other)	parameter other.time을 받아 this.time을 빼서 시간의 차를 계산하는 메소드	

- **ParkingCenter class**

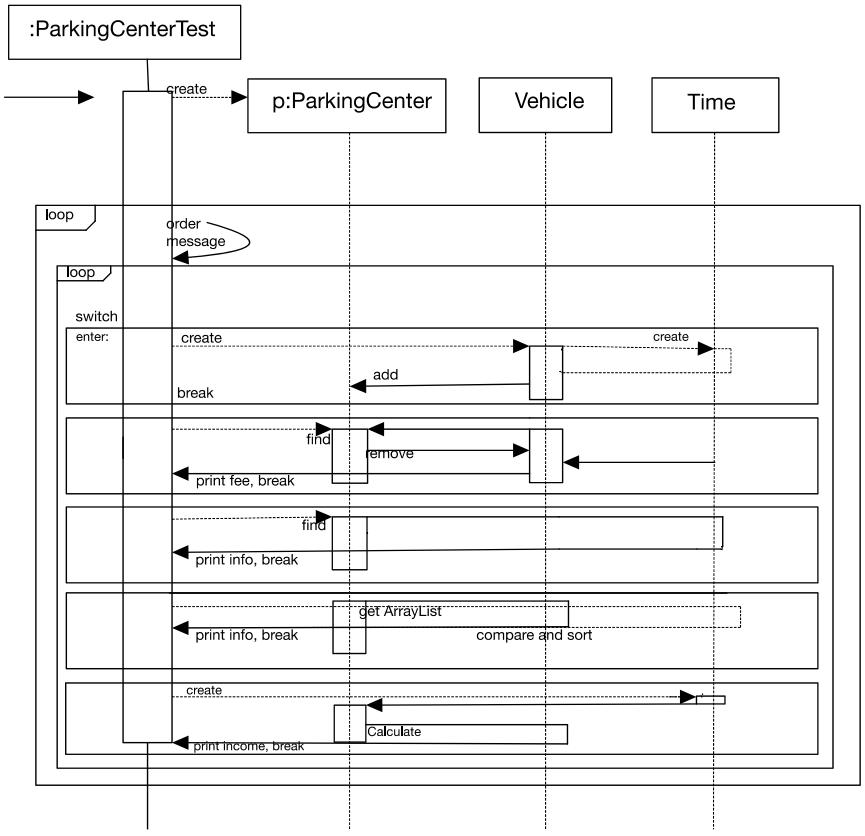
Method	설명	Data
incar(Vehicle v)	주차차량 ArrayList에 차량을 추가	-
outcar(Vehicle v)	주차차량 ArrayList에서 차량을 삭제하고 출차차량 ArrayList에 차량을 추가	-
reincar(Vehicle v)	나갔다 다시 들어오는 차량일 시 사용, 출차차량 ArrayList에서 차량을 삭제, 주자차량 ArrayList에 차량 추가	-

addleave(Vehicle v)	leavelist에 차량 추가	-
findcar(int number)	int형 정수를 받아들여 주차차량 ArrayList를 for문을 통해 반복하여 비교하며 같은 차량번호가 있다면 k를 반환함.	k: 같은 번호 있으면 1, 없으면 0이다.
isreturncar(int number)	int형 정수를 받아들여 출차차량 ArrayList를 for문을 통해 반복하여 비교하며 같은 차량번호가 있다면 k를 반환함.	k: 같은 번호 있으면 1, 없으면 0이다.
get_index()	index를 반환함.	-
getcar(int index)	index를 받아 그에 맞는 주차차량을 반환한다.	-
getreturncar(int index)	index를 받아 그에 맞는 출차차량을 반환한다.	-
How_Many_Cars()	주차차량이 몇 대가 있는지 반환한다.	-
show(Time t, Vehicle c)	받아들인 차량 정보를 보여주는 메소드	
list(Time t)	특정 시간의 주차 리스트를 보여주고, 전기차의 경우 특정 시간에 해당하는 충전량을 보여주는 메소드	-
income(String s)	leavelist의 차량을 모두 불러와 해당날짜와 같은 출차날짜를 가진 차량의 요금을 모두 더하는 메소드	k : 요금이 저장되는 변수

4) Activity Diagram

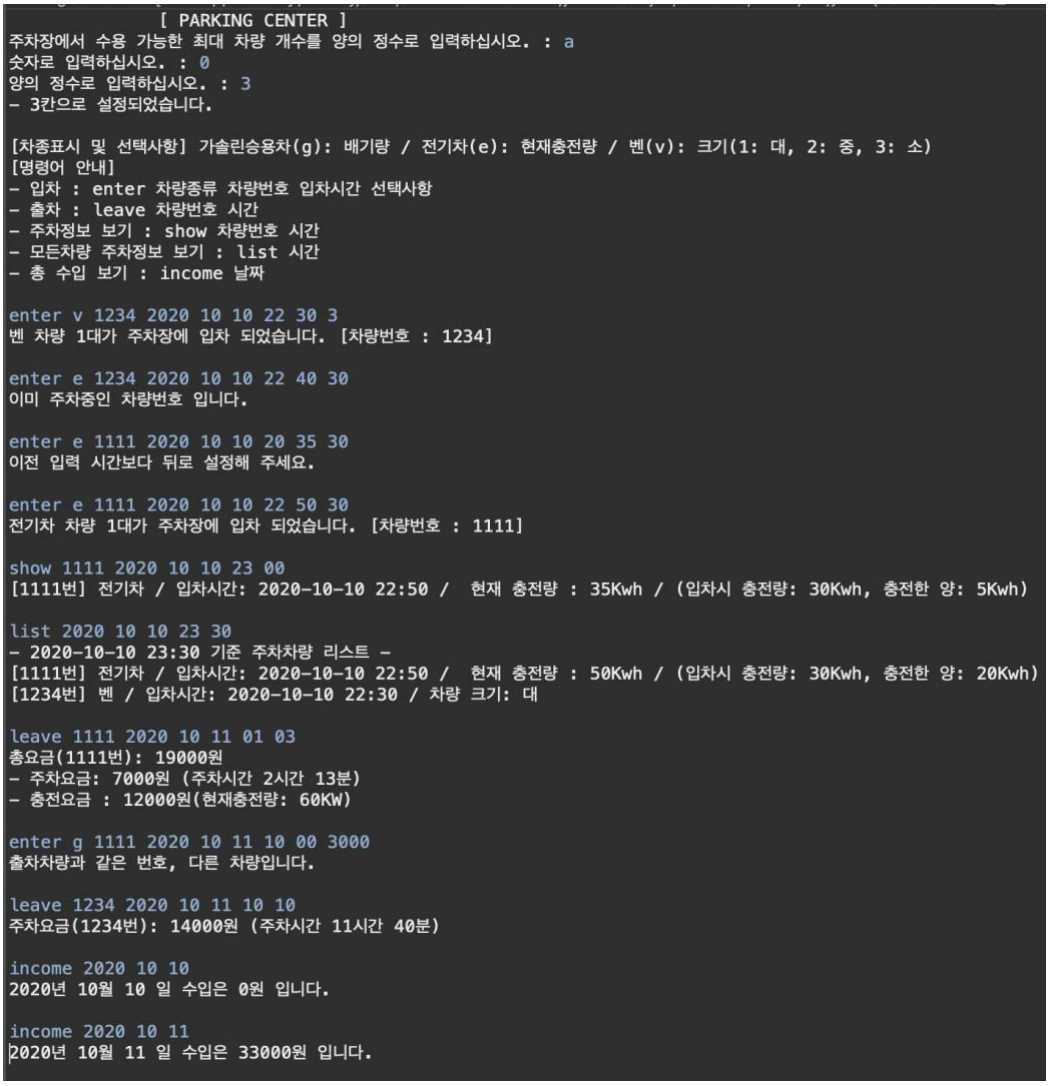


5) Sequence Diagram



3. 출력 결과 및 설명

TEST1)



```
                [ PARKING CENTER ]
주차장에서 수용 가능한 최대 차량 개수를 양의 정수로 입력하십시오. : a
숫자로 입력하십시오. : 0
양의 정수로 입력하십시오. : 3
- 3칸으로 설정되었습니다.

[차종표시 및 선택사항] 가솔린승용차(g): 배기량 / 전기차(e): 현재충전량 / 벤(v): 크기(1: 대, 2: 중, 3: 소)
[명령어 안내]
- 입차 : enter 차량종류 차량번호 입차시간 선택사항
- 출차 : leave 차량번호 시간
- 주차정보 보기 : show 차량번호 시간
- 모든차량 주차정보 보기 : list 시간
- 총 수입 보기 : income 날짜

enter v 1234 2020 10 10 22 30 3
벤 차량 1대가 주차장에 입차 되었습니다. [차량번호 : 1234]

enter e 1234 2020 10 10 22 40 30
이미 주차중인 차량번호 입니다.

enter e 1111 2020 10 10 20 35 30
이전 입력 시간보다 뒤로 설정해 주세요.

enter e 1111 2020 10 10 22 50 30
전기차 차량 1대가 주차장에 입차 되었습니다. [차량번호 : 1111]

show 1111 2020 10 10 23 00
[1111번] 전기차 / 입차시간: 2020-10-10 22:50 /  현재 충전량 : 35Kwh / (입차시 충전량: 30Kwh, 충전한 양: 5Kwh)

list 2020 10 10 23 30
- 2020-10-10 23:30 기준 주차차량 리스트 -
[1111번] 전기차 / 입차시간: 2020-10-10 22:50 /  현재 충전량 : 50Kwh / (입차시 충전량: 30Kwh, 충전한 양: 20Kwh)
[1234번] 벤 / 입차시간: 2020-10-10 22:30 / 차량 크기: 대

leave 1111 2020 10 11 01 03
총요금(1111번): 19000원
- 주차요금: 7000원 (주차시간 2시간 13분)
- 충전요금 : 12000원(현재충전량: 60KW)

enter g 1111 2020 10 11 10 00 3000
출차차량과 같은 번호, 다른 차량입니다.

leave 1234 2020 10 11 10 10
주차요금(1234번): 14000원 (주차시간 11시간 40분)

income 2020 10 10
2020년 10월 10 일 수입은 0원 입니다.

income 2020 10 11
2020년 10월 11 일 수입은 33000원 입니다.
```

- 먼저 실행 시 주차칸 개수를 설정하는데 위와 같이 문자나 1~9999가 아닌 수를 입력하면 오류가 발생한다. enter를 통해 1234번 벤 차량이 입차하고, 1234번 전기차가 입차하려 하자 주차차량 번호라고 오류가 발생한다. 1111번 전기차가 이전 명령보다 이른 시간에 입차하려 하자 오류가 발생한다. show 명령어를 통해 1111번 전기차의 차량정보와 입력한 시간에 대한 충전량을 출력한다. 이후 list 명령어를 통해 주차한 모든 차량을 보여 준다. 1111번 전기차의 경우 list 명령이 show 명령보다 20분이 증가한 시간이기 때문에 충전량이 더 늘어 출력된다. leave 명령을 통해 1111번 전기차가 출차하는데, 주차시간이 2시간 13분임에도 충전요금은 30Kwh를 충전하는 금액 12000원에서 더 오르지 않음을 알 수 있다. 이후 1111번 가솔린 차량이 입차하려 하자 출차한 1111번과 다른 차량임을 잡아내고 오류가 발생한다. 이후 leave를 통해 1234번 벤이 출차한다. 출차한 1111번 전기차와 1234번 벤은 10월 10일에 입차하여 10월 11일에 출차하였다. Income 명령을 통해 10월 10일과 10월 11일을 둘다 출력한 결과 11일(출차날짜)에 수입이 저장되었음을 알 수 있다.

TEST2)

```
                [ PARKING CENTER ]
주차장에서 수용 가능한 최대 차량 개수를 양의 정수로 입력하십시오. : 4
- 4칸으로 설정되었습니다.

[차종표시 및 선택사항] 가솔린승용차(g): 배기량 / 전기차(e): 현재충전량 / 벤(v): 크기(1: 대, 2: 중, 3: 소)
[명령어 안내]
- 입차 : enter 차량종류 차량번호 입차시간 선택사항
- 출차 : leave 차량번호 시간
- 주차정보 보기 : show 차량번호 시간
- 모든차량 주차정보 보기 : list 시간
- 총 수입 보기 : income 날짜

enter v 1111 2021 05 22 07 18 4
벤의 크기는 1 ~ 3 사이여야 합니다.

enter v 1111 2021 05 22 07 18 1
벤 차량 1대가 주차장에 입차 되었습니다. [차량번호 : 1111]

enter e 2222 2021 05 32 07 20 30
유효하지 않은 날짜입니다.

enter e 2222 2021 05 22 07 20 30
전기차 차량 1대가 주차장에 입차 되었습니다. [차량번호 : 2222]

enter e 3333 2021 05 22 07 25 61
전기차의 충전량은 1 ~ 60 사이여야 합니다.

enter e 3333 2021 05 22 07 25 60
전기차 차량 1대가 주차장에 입차 되었습니다. [차량번호 : 3333]

enter g 4444 2021 05 22 07 26 1000
일반승용차 차량 1대가 주차장에 입차 되었습니다. [차량번호 : 4444]

enter g 5555 2021 05 22 07 30 500
주차 공간이 부족합니다.

list 2021 05 22 08 00
- 2021-05-22 08:00 기준 주차차량 리스트 -
[4444번] 일반승용차 / 입차시간: 2021-05-22 07:26 / 배기량: 1000cc
[2222번] 전기차 / 입차시간: 2021-05-22 07:20 /  현재 충전량 : 50Kwh / (입차시 충전량: 30Kwh, 충전한 양: 20Kwh)
[3333번] 전기차 / 입차시간: 2021-05-22 07:25 /  현재 충전량 : 60Kwh / (입차시 충전량: 60Kwh, 충전한 양: 0Kwh)
[1111번] 벤 / 입차시간: 2021-05-22 07:18 / 차량 크기: 대

leave 4444 2021 05 22 07 20
출차시간을 입차시간보다 뒤로 설정해 주십시오.

leave 4444 2021 05 22 08 10
주차요금(4444번): 1000원 (주차시간 0시간 44분)
```

- 1111번 벤이 입차하려 하지만, 벤의 크기 입력 오류가 발생한다. 다시 크기가 1인 1111번 벤이 입차한다. 2222번 전기차가 입차하려 하지만 날짜가 2021/05/32 이므로 유효성검사에서 오류가 발생한다. 다시 2222번 전기차가 입차한다. 3333번 전기차가 입차하려 하지만 충전량이 61이라 입력 오류가 발생한다. 다시 3333 전기차가 입차한다. 그리고 4444번 일반승용차가 입차한다. 이로써 총 4대의 차가 입차하였는데, 5555번 경차가 입차하려 한다. 허나 주차 공간을 4칸으로 설정했으므로 주차 공간 부족 메세지가 출력되며 입차되지 않는다. 리스트를 통해 4대의 주차중인 차량정보를 출력한다. 일반승용차 -> 전기차 -> 벤 순서대로 나열되고, 같은 차종에서는 입차시간을 기준으로 나열된다. 그리고 4444번 일반승용차가 출차하려 하지만 입차시간(07:26)보다 이른 출차시간(07:20)을 입력하여 오류가 발생한다. 그리고 나서 알맞는 시간에 4444번 일반승용차는 출차한다.

4. 결론

- 또 한번의 객체지향프로그래밍 수업의 커다란 과제가 끝이 났다. 첫 번째 과제보다 분명 어려웠다고 느끼지만, 그 당시보다는 꾸준히 공부하여 성장하여 문맹과 같던 상태를 벗어났다고 자부했기에 전보다 짧은 시간인 3일을 잡고 시작했지만 실수였다.

미리 완벽한 틀을 짜 놓고 시작할 실력이 아닌 나는 즉석으로 방향과 해결책을 생각해내며 하나씩 기능을 늘려 갔기 때문에, 불필요한 코드나 변수가 존재하거나, 쉽게 갈 길을 빙빙 돌아갔을 수도 있다고 느꼈다. 후반부에서는 그래서인지 더 오류가 많이 발생하였다. 가장 어려웠던 부분은, income 명령을 구현하는 것이었다. ArrayList를 하나 더 만들어 출차차량을 추가시키고, 날짜 + 시간 정보에서 날짜만으로 비교할 수 있도록 하여 해결하였다. 그리고 오류처리를 할 것이 정말 많아 조금 어려움을 겪었다. 특히 명령 간 시간 순서를 검사하는 것과 날짜의 유효성 검사를 어렵게 느꼈다.

과제를 통하여 많은 것을 배웠는데, 상속과 다형성에 대해 보다 더 이해할 수 있었으며, 오류처리의 중요성을 알게 되었다. 거의 나흘을 밤새워 했음에도 실력이 부족하여 시간이 오래 걸렸고 어려움이 많았다. 그렇지만 오기와 즐거움이 공존한 시간이었기에 완성되었을 때의 보람은 이루 말할 수 없고, 더 열정적으로 공부하고 시도해야겠다는 생각이 든다.