2020년 2학기
객체지향프로그래밍 및 실습
2차 프로그래밍 과제

HW2

수학과 201921195

2학년 조우연

[1] 서론

본 보고서는 2020학년 2학기 객체지향프로그래밍 및 실습 과목의 2차 프로그래밍 과제에 대한 보고서로서 승용차, 버스, 트럭 세 가지 종류의 차량이 주차 가능한 주차장 관리 프로그램에 대한 보고서이다. 주요 데이터를 클래스화 하고, 상속과 다형성을 활용하여 객체지향적 프로그래밍 방법으로 프로그램을 설계하고 구현하는 것을 핵심으로 한다.

2차 프로그래밍 과제에 대한 문제요구사항은 과제 문제지에 주어진 것과 같으며 문제요구사항에서 언급된 기능은 모두 구현되었음을 명시한다.

- 소개(구현)

class	instance variable	constructor	method
Car	num:int in:Time out:Time cost:int now:boolean duration:int	Car(int, Time) Car()	getNum():int getIn():Time getOut():Time getCost():int getNow():boolean getDuration():int setDuration(int):void setOut(int,int,int,int,int):void outNow():void CalCost():int
Time	year:int month:int day:int hour:int min:int	Time(int,int,int,int,int)	getYear():int getMonth():int getDay():int getHour():int getMin():int setTime(int,int,int,int,int):void CalTime(Time):int Static method isValid(int,int,int,int,int):boolean isValid2(TIme,Time):boolean
ParkingLot	size:int Cars:Car[] count:int COST:int	-	Static method Init():void In():void Out():void parkingList():void AllCost():void compare(Car,Car):int check():int

Car의 subclass			
subclass	**(추가된) instance variable**	**constructor**	**method**
Normal	-	Normal(int,TIme) Normal()	CalCost():int
Bus	-	Bus(int,Time) Bus()	-
Truck	-	Truck(int,Time) Truck()	-

Normal의 subclass			
subclass	**(추가된) instance variable**	**constructor**	**method**
Electric	battery:double	Electric(int,Time,double)	CalCost():int

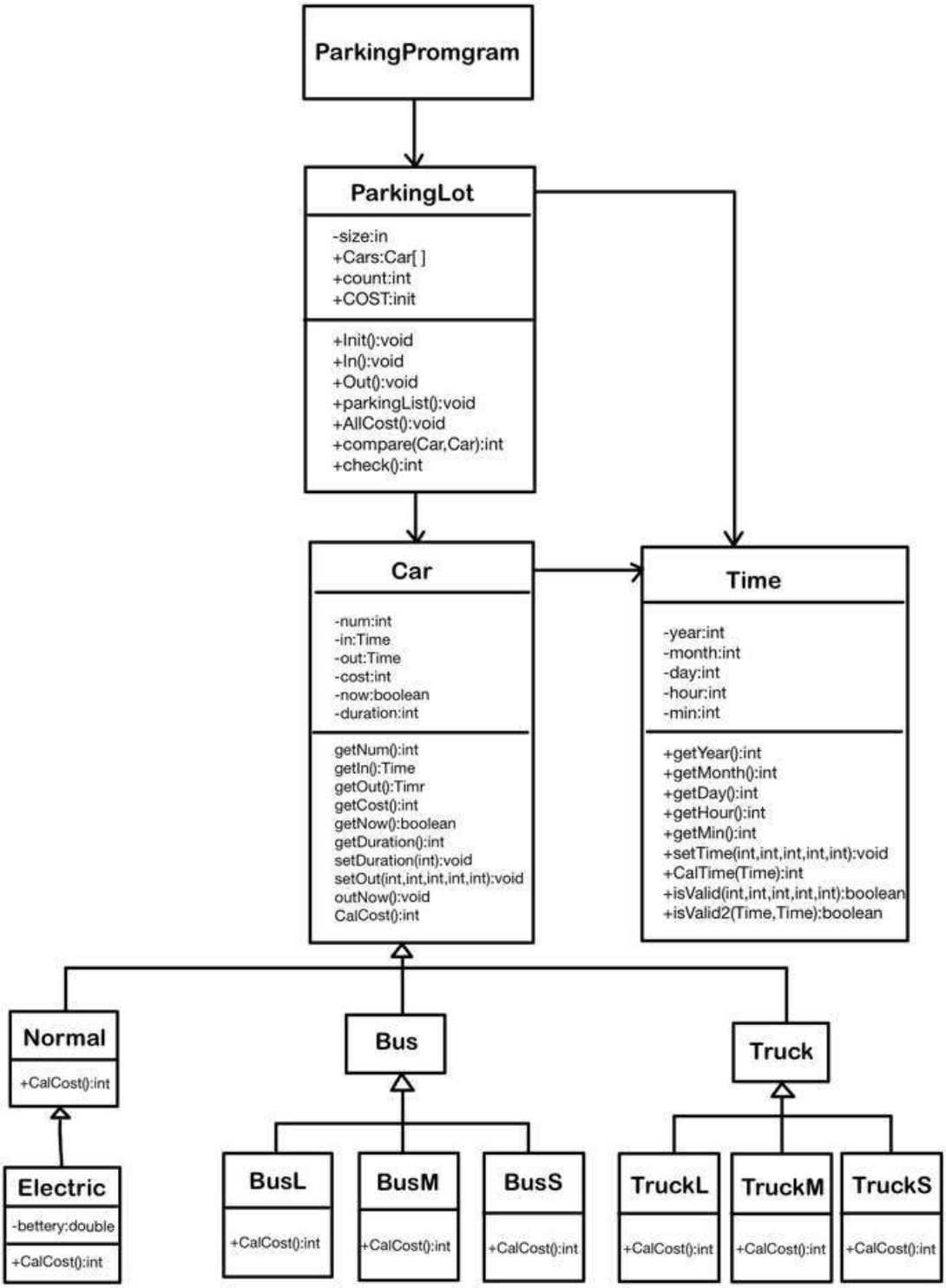
Bus의 subclass			
subclass	**(추가된) instance variable**	**constructor**	**method**
BusL	-	BusL(int,Time)	CalCost():int
BusM	-	BusM(int,Time)	CalCost():int
BusS	-	BusS(int,Time)	CalCost():int

Truck의 subclass			
subclass	(추가된) instance variable	constructor	method
TruckL	-	TruckL(int,Time)	CalCost():int
TruckM	-	TruckM(int,Time)	CalCost():int
TruckS	-	TruckS(int,Time)	CalCost():int

[2] 본론

[가] 분석/설계

- UML class diagram



- 주요 instance variable 및 메소드 설명

Car 클래스의 instance variable	Robot 클래스의 메소드
private num : int ->차량번호를 저장한다. in : Time ->입차시간을 저장한다. out : Time ->출차시간을 저장한다. cost : int ->주차비용을 저장한다. now : boolean ->현재 주차되어있는지의 여부를 저장한다. 주차중일 때 true, 주차중이 아닐 때(출차되었을 때) false로 저장한다. duration : int ->입차 후 출차한 시간의 차이를 저장한다. 분 단위로 저장한다.	Duration() : int ->좌표를 매개변수로 받아, 로봇의 위치를 넘겨받은 좌표로 pos를 변경한다. 이때 이동하기 전의 좌표의 값은 board의 method인 origin()을 통해 0으로 바꾸고, 이동후의 좌표의 값은 board의 record() 메소드를 통해 1로 바꾼다. getNum(), getCost(), getDuration() : int ->각각 num, cost, duration을 int형으로 반환한다. getIn(), getOut() : Time ->각각 in, out을 Time형태로 반환한다. getNow() : boolean ->now를 boolean형태로 반환한다. setDuration(int) : void ->duration을 넘겨받은 값으로 바꾼다. setOut(int,int,int,int,int) : void ->Time형태의 Out을 주어진 날짜정보로 세팅한다. outNow() : void ->now를 false로 바꾼다. CalCost() : int ->default method(Overriding 위함)

Car의 subclass의 메소드	
CalCost() : int ->Car의 subclass타입에 따라 각각에 기준에 맞게 주차요금을 계산하여 반환한다. 차량 종류(Car의 subtype)에 따라 주차요금이 달라지므로 각각에 부합하게 overriding한다. => Car의 subclass들은 super class의 instance variable들과 method들을 상속받으며 CalCost() method는 각각의 type에 맞게 overriding되고, Electric class만 유일하게 새로운 instance variable과 method가 정의되어 있으며 이에 대한 내용은 아래와 같다.	
Electic클래스에 추가된 instance variable	**Electic클래스의 method**
private	setBattery(double) : void

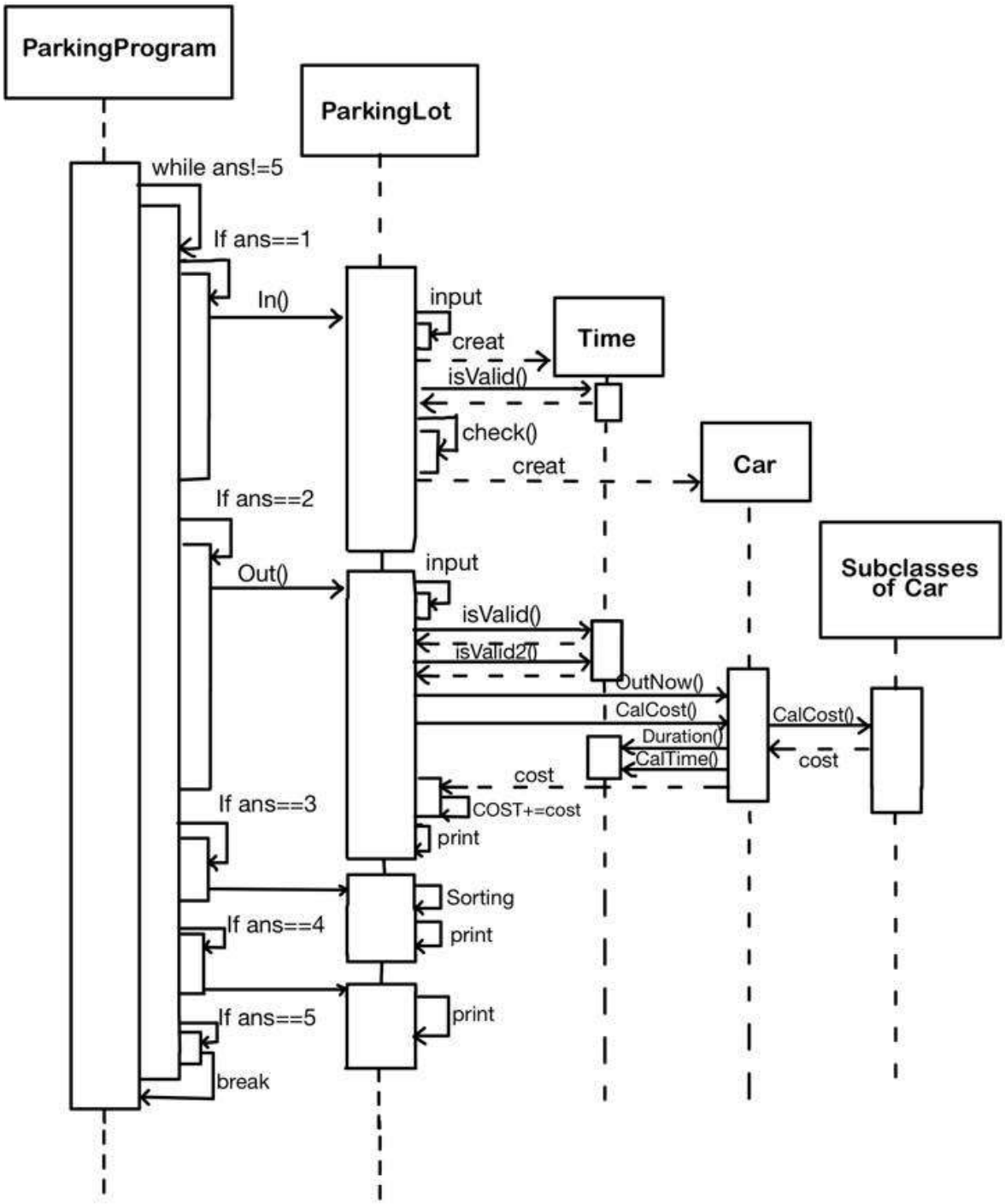
battery : double -> 전기차의 배터리잔여량을 저장한다.	->battery의 값을 넘겨받은 값으로 바꾼다. CalCost() : int ->주차요금과 배터리충전요금을 계산하고 두 값을 더한 값을 반환한다.

Time 클래스의 instance variable	Time 클래스의 메소드
private year : int ->년도를 저장한다. month : int ->월을 저장한다. day : int ->일을 저장한다. hour : int ->시간을 저장한다. min : int ->분을 저장한다.	setTime(int,int,int,int,int) : void ->주어진 정보로 year, month, day, hour, min을 바꾼다. getYear(), getMonth(), getDay(), getHour(), getMin() : int ->각각 연, 월, 일, 시, 분을 int형으로 반환한다. CalTime(Time) : int -> 두 시간의 차이를 계산해 분단위로 반환한다. static method isValid(int,int,int,int,int) : boolean ->넘겨받은 정보가(각각 년,월,일,시,분)이 유효한 범위 안에 있는지 확인한다. 유효한 시간이면 true, 그렇지 않으면 false를 반환한다. isValid2(int,int,int,int,int) : boolean ->두 시간차이(입차,출차시간)이 올바른지 검사한다. 입차시간이 출차시간보다 빠르면 true, 늦으면 false를 반환한다.

ParkingLot 클래스의 instance variable	ParkingLot 클래스의 메소드
public static variables size : int ->최대 주차가능 차량 수 Cars : Car[] ->배열(=주차장)에 차량정보 저장 count ->주차장에 주차되어 있는 차량 수 COST ->총 수입을 저장한다.	static method In() : void ->입차할 차량종류와 차량번호, 입차시간을 입력받아 차량종류에 맞는 객체를 생성하고, Cars(주차장)에 저장한다. Out() : void ->출차할 차량번호와 출차시간을 입력받아 주차장에서 출차시킨다. 즉, 출차할 car의 now를 false로 변환시키고, 주차요금을 계산해 출력하고, 총수입에 정산된 주차요금을 반영한다. parkingList() : void ->주차장 즉, Cars에 저장되어있는 차량정보를 각각의 차량 종류에 따라 분류하고, 입차시간 순서별로 sorting하여 주차되어 있는 모든 차량의 종류별 입차시간 순서로 차량정보를 출력한다. AllCost() : void ->총수입을 출력한다. compare(Car,Car) : int ->두 차량의 입차시간을 비교한다. 넘겨받은 Car타입의 변수 a,b중 a의 입차시간이 더 늦으면 1, 더 빠르면 -1을 반환하고, 두 차량의 입차시간이 같으면 0을 반환한다. check() : int ->주차장(Cars배열)의 빈자리를 찾아 빈자리의 index를 반환한다. 빈자리가 없으면 -1을 반환한다.

[나] 2차 프로그래밍 과제에 대한 sequence diagram

(본 sequence diagram은 본 코드의 주요 알고리즘 및 주요 메소드 호출에 대해서 작성되었습니다.)



ParkingProgram class의 main에서 시작한다. main에서 while문을 돌며 사용자에게 사용할 기능의 메뉴를 입력받고, 입력받은 기능과 관련한 static method를 호출해 실행하는 형태로 실행된다.

Car의 subclass는 대부분 CalCost() method를 overriding한 메소드만 가지고 있으므로 하나로 묶어서 작성하였다.(전기차에 관한 class는 setBattery에 대한 method가 추가로 있지만, 프로그램 전체 틀에 있어서 주요 알고리즘에 큰 영향을 미치지 않으므로 다른 subclass들과 함께 표현하였다.)

[다] ParkingProgram(main 함수가 있는 클래스) 실행결과화면, 결과에 대한 설명

1. 정상실행결과

입차

```
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
1
차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b)
c 0
차량 번호를 입력하세요! (4자리 숫자)
1111
입차시간을 입력하세요! (년 월 일 시 분)
2014 5 2 10 30
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
1
차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b)
c 5
차량 번호를 입력하세요! (4자리 숫자)
2222
입차시간을 입력하세요! (년 월 일 시 분)
2018 7 6 10 30
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
1
차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b)
t 10
차량 번호를 입력하세요! (4자리 숫자)
3333
입차시간을 입력하세요! (년 월 일 시 분)
2020 10 2 5 5
```

```
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
1
차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b)
t 7
차량 번호를 입력하세요! (4자리 숫자)
4444
입차시간을 입력하세요! (년 월 일 시 분)
2020 8 1 13 30
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
1
차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b)
t 3
차량 번호를 입력하세요! (4자리 숫자)
5555
입차시간을 입력하세요! (년 월 일 시 분)
2020 12 6 15 0
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
1
차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b)
b 10
차량 번호를 입력하세요! (4자리 숫자)
6666
입차시간을 입력하세요! (년 월 일 시 분)
2020 3 16 9 05
```

```
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
1
차량 종류 및 용량을 입력하세요! 승용차(c), 트럭(t), 버스(b)
b 30
차량 번호를 입력하세요! (4자리 숫자)
7777
입차시간을 입력하세요! (년 월 일 시 분)
2020 2 6 10 10
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
1
차량 종류 및 용량을 입력하세요! 승용차(c), 트럭(t), 버스(b)
b 45
차량 번호를 입력하세요! (4자리 숫자)
8888
입차시간을 입력하세요! (년 월 일 시 분)
2021 5 8 10 0
```

위 실행(차량 종류별 입차) 후 주차차량 보기	입차만 실행 후 총 수입보기
원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 3 승용차 1111 2014/05/02 10:30 승용차 2222 2018/07/06 10:30 버스 7777 2020/02/06 10:10 버스 6666 2020/03/16 09:05 버스 8888 2021/05/08 10:00 트럭 4444 2020/08/01 13:30 트럭 3333 2020/10/02 05:05 트럭 5555 2020/12/06 15:00	원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 4 총 수입은 0원 입니다.

출차(일부만)

```
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
2
출차할 차량번호를 입력하세요!
1111
출차시간을 입력하세요!
2014 5 2 12 45
주차시간은 2시간 20분입니다.
주차요금은 6,500원입니다.

원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
2
출차할 차량번호를 입력하세요!
2222
출차시간을 입력하세요!
2018 7 6 12 45
주차시간은 2시간 20분입니다.
주차요금은 14,600원입니다.

원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
2
출차할 차량번호를 입력하세요!
3333
출차시간을 입력하세요!
2020 10 2 7 42
주차시간은 3시간 0분입니다.
주차요금은 12,000원입니다.
```

```
원하는 기능을 선택하세요!
1. 입차
2. 출차
3. 주차차량 보기
4. 총 수입 보기
5. 종료
2
출차할 차량번호를 입력하세요!
6666
출차시간을 입력하세요!
2020 3 16 13 0
주차시간은 4시간 0분입니다.
주차요금은 8,000원입니다.
```

위 실행(일부 출차) 후 주차차량 보기	일부 출차 실행 후 총 수입보기
원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 3 버스 7777 2020/02/06 10:10 버스 8888 2021/05/08 10:00 트럭 4444 2020/08/01 13:30 트럭 5555 2020/12/06 15:00	원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 4 총 수입은 41,100원 입니다.

출차(나머지 차량)	
원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 2 출차할 차량번호를 입력하세요! 7777 출차시간을 입력하세요! (년 월 일 시 분) 2020 2 7 10 0 주차시간은 24시간 0분입니다. 주차요금은 72,000원입니다. 원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 2 출차할 차량번호를 입력하세요! 5555 출차시간을 입력하세요! 2020 12 6 23 10 주차시간은 9시간 0분입니다. 주차요금은 18,000원입니다.	원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 2 출차할 차량번호를 입력하세요! 8888 출차시간을 입력하세요! 2021 5 8 15 01 주차시간은 5시간 30분입니다. 주차요금은 22,000원입니다. 원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 2 출차할 차량번호를 입력하세요! 4444 출차시간을 입력하세요! 2020 8 1 15 20 주차시간은 2시간 0분입니다. 주차요금은 6,000원입니다.

위 실행(나머지 모든 출차) 후 주차차량 보기	모든 차량 출차 실행 후 총 수입보기
원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 3 주차된 차량이 없습니다.	원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 4 총 수입은 159,100원 입니다.

종료 실행
원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 5 프로그램을 종료합니다.

실행결과에 대한 설명

- 입차

"입차"를 실행 시, 차량의 입차정보(차량종류, 차량종류, 입차시간)을 입력받아 저장한다. 위 실행결과는 다양한 차량이 입차하는 경우를 다루기 위해 차량의 종류를 모두 다르게 해서 프로그램을 실행한 결과이다. 즉, 각각 **일반승용차, 전기차, 대형/중형/소형 트럭 및 대형/중형/소형 버스에 대한 테스트 케이스를 모두 다룬다.**

- 출차

"출차"를 실행 시, 출차하는 차량의 정보(차량번호, 출차시간)을 입력받아 출차하는 차량의 총 주차요금을 계산하여 출력한다. **자동차 종류에 따라 계산되는 요금이 다르므로**, 각각에 대한 요금계산도 반영된 결과를 볼 수 있다. 위 프로그램에서 계산되어 출력된 차량별 주차요금은 아래의 표와 같다.

차량 종류	중량or최대 승객수or배터리잔량	차량 번호	입차시간	출차시간	주차시간	요금
c	0	1111	2014.5.2.10:30	2014.5.2.12:45	2시간20분	1000+11 500=6500
c	5	2222	2018.7.6.10:30	2018.7.6.12:45	2시간20분	6500+0.2×135×300=14600(주차+충전)
t	10	3333	2020.10.2.5:5	2020.10.2.7:42	3시간	4000×3=12000
t	7	4444	2020.8.1.13:30	2020.8.1.15:20	2시간	3000×2=6000

t	3	5555	2020.12.6.15:00	2020.12.6.23:10	9시간	2000 9=18000
b	10	6666	2020.3.16.9:5	2020.3.16.13:00	4시간	2000×4=8000
b	30	7777	2020.2.6.10:10	2020.2.7.10:00	24시간	3000×24=72000
b	45	8888	2021.5.8.10:00	2021.5.8.15:01	5시간30분	4000×5.5=22000

- 주차차량보기

"주차차량보기"를 실행하면 주차장에 남아있는 차량들을 승용차->버스->트럭 순으로, 또한 입차시간 순서대로 출력된다. 모든 경우를 다루기 위해서, **입차만 실행된 경우, 입차 후 일부 차량만 출차가 실행된 경우, 모든 차량의 출차가 실행된 경우**로 나누어 프로그램을 실행시켰다.

- 총 수익보기

"총 수입보기"를 실행하면 입차 후 출차한 차량들의 요금의 합을 출력한다. 이도 마찬가지로 모든 경우에 따라서 총 수입이 잘 반영되었는지 확인하기 위해 **입차만 실행된 경우, 입차 후 일부 차량만 출차가 실행된 경우, 모든 차량의 출차가 실행된 경우**로 나누어 각각에 대한 총 수입이 잘 반영되었는지를 체크하였다.

- 종료

"종료"를 선택하면 프로그램이 종료된다.

2. 예외처리

유효성 검사 및 오류 검사(예외처리)	
잘못된 차량종류를 입력하는 경우	유효하지 않는 날짜를 입력할 경우
차량 종류 및 용량을 입력하세요! 승용차(c), 트럭(t), 버스(b) d 0 존재하지 않는 차량 종류를 입력하였습니다. 차량 종류 및 용량을 입력하세요! 승용차(c), 트럭(t), 버스(b)	입차시간을 입력하세요! (년 월 일 시 분) 2021 4 31 15 20 잘못된 시간(존재하지 않는 시간or날짜)을 입력하였습니다. 올바른 시간을 입력해 주세요. 입차시간을 입력하세요! (년 월 일 시 분) 2021 4 30 5 67 잘못된 시간(존재하지 않는 시간or날짜)을 입력하였습니다. 올바른 시간을 입력해 주세요. 입차시간을 입력하세요! (년 월 일 시 분)
입차시간이 출차시간보다 늦는 경우	차량종류와 함께 용량(중량, 최대승객수, 배터리잔량)을 입력할 때 용량이 터무니 없는 숫자일 경우 (매우 큰 수나 음수)

차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b) c 0 차량 번호를 입력하세요! (4자리 숫자) 1234 입차시간을 입력하세요! (년 월 일 시 분) 2020 5 5 12 10 원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 2 출차할 차량번호를 입력하세요! 1234 출차시간을 입력하세요! 2020 5 5 10 15 출차시간이 입차시간보다 빠릅니다. 올바른 출차시간을 입력해 주세요. 출차시간을 입력하세요! (년 월 일 시 분)	차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b) t 70 출입가능한 트럭의 최대중량은 20입니다. 차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b) b 100 출입가능한 버스의 최대 승객수는 60명입니다. 차량 종류 및 용량을 입력하세요! 승용자(c), 트럭(t), 버스(b) c -5 전기차의 배터리 잔량은 0보다 작을 수 없습니다.
출차 차량번호가 주차 차량 목록에 없는 경우	메뉴 번호 범위를 잘못 입력하는 경우
출차할 차량번호를 입력하세요! 1111 존재하지 않는 차량번호입니다. 출차할 차량번호를 입력하세요!	원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료 6 잘못된 번호를 입력하였습니다. 1-5번 사이의 메뉴를 선택해 주세요. 원하는 기능을 선택하세요! 1. 입차 2. 출차 3. 주차차량 보기 4. 총 수입 보기 5. 종료

[3] 결론

2차 프로그래밍 과제를 수행함으로써 객체지향적 특징을 잘 활용하고, 클래스의 상속의 관계에 대해서 명확하게 이해하고 활용할 수 있게 되었다. 또한 클래스의 상속이 코드를 더욱 간결하게 하고, 상속관계에서 얻을 수 있는 다형성이 코드를 더욱 일반화하여 작성할 수 있게 해준다는 것을 경험하였다. 특히 overriding을 통해 상황에 맞게 method를 재정의함으로써 활용도를 높인다는 등의 장점도 직접 경험하게 되었다.

1차 프로그래밍 과제와 비교했을 때, 1차 프로그래밍 과제는 객체지향적인 프로그래밍보다는 구조적 프로그래밍에 가깝게 코딩을 했지만, 2차 프로그래밍 과제에서는 더욱 객체지향적인 프로그래밍을 할 수 있게 되었다. 구조적으로 프로그래밍을 했을 때에는 메소드 호출 관계나 책임분배 등이 깔끔하게 잘 이루어지지 않아 설계가 더욱 복잡하고, 코드도 산만하게 작성되었는데, 객체지향 프로그래밍을 적용하여 코딩을 한 결과, 설계도 더욱 깔끔해졌고, 코드 가독성도 더 개선된 것으로 보인다. 무엇보다도 1차 프로그래밍 과제에 비해 encapsulation을 깨뜨리지 않게끔 안정적으로 코딩을 할 수 있었다. 이전에는 get method, set method등을 사용하지 않고, (예를들어) Employee.salary 과 같이 instance variable에 접근하였는데 이번 과제에서는 class의 instance variables를 private으로 선언하고 get, set method를 잘 활용하여 instance variable들의 값을 안전하게 보호하고, encapsulation을 위반하지 않도록 프로그래밍을 진행하였다.

예외처리는 생각보다 까다로운 부분이 많았다. 일상생활에서 당연하게 전제되는 것들이 프로그램 내에서는 당연하게 내제되어있지 않기 때문에, 필요한 예외나 유효성 검사 등을 일일이 해주어야 해서 번거롭기도 하고, 복잡하게 체감되었다. 특히나 날짜와 관련된 유효성 검사는 상황마다 유효한 날짜가 다르기 때문에 이를 나누어 예외를 처리하는데 어려움이 있었다. 날짜, 시간에 대한 것은 예외처리나 유효성 검사 이외에도 번거로운 것들이 꽤 많았는데, 특히나 날짜, 시간 차이에 대한 것이 그러했다. 년, 월, 일, 시, 분이 모두 주어져 있을 때 두 시간의 차이를 계산하는 것은 매우 어려웠다. 단순 뺄셈의 개념이 아닌, 년도, 달, 일, 시, 분의 단위가 모두 달라 이를 환산하는 것이 굉장히 복잡하기 때문이다. 이러한 것들을 직접 구현한 것은 프로그래밍을 하는 것과 계산에 대해서 좋은 연습과 경험이 되었지만, 이와 관련된 모듈을 찾아서 사용하는 연습도 필요할 것 같다.