

CSED101. Programming & Problem solving

Spring 2023

Programming Assignment #3 (80 points)

김휘 (hwikim@postech.ac.kr)

■ 제출 마감일: 2023.05.22 23:59

■ 파이썬 버전: Python 3.x

■ 제출물

- .py 소스 코드 (assn3.py)
 - 프로그램의 소스 코드에 채점자의 이해를 돕기 위한 주석을 반드시 붙여주세요.
- 보고서 파일 (.docx, .hwp 또는 .pdf; assn3.docx, assn3.hwp 또는 assn3.pdf)
 - 보고서는 AssnReadMe.pdf를 참조하여 작성하시면 됩니다.
 - 명예 서약 (Honor code): 표지에 다음의 서약을 기입하여 제출해 주세요: “나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.” 보고서 표지에 명예 서약이 기입되어 있지 않은 과제는 제출되지 않은 것으로 처리됩니다.
 - 작성한 소스 코드와 보고서 파일은 PLMS를 통해 제출해 주세요.

■ 주의 사항

- 구문 오류(Syntax Error)가 발생하거나 실행이 되지 않는 과제는 0점으로 채점됩니다.
- 제출 기한보다 하루 늦게 제출된 과제는 최종 20%, 이틀 늦게 제출된 과제는 최종 40% 감점됩니다. 제출 기한보다 사흘 이상 늦으면 제출 받지 않습니다 (0점 처리).
늦은 제출시 PLMS에 기록된 최종 수정일시를 기준으로 감점합니다.
- 각 문제의 제한 조건과 요구 사항을 반드시 지켜 주시기 바랍니다.
- 모든 문제의 출력 형식은 채점을 위해 아래에 제시된 예시들과 최대한 비슷하게 작성해 주세요.
- 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.
- 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 “POSTECH 전자컴퓨터공학부 부정행위 정의”를 따릅니다 (PLMS의 본 과목 공지사항에 등록된 글 중, 제목이 [document about cheating]인 글에 첨부되어 있는 disciplinary.pdf를 참조하세요).

■ Problem: 주유소 관리 (Gas Station Management)

[문제]

주유소를 관리하는 프로그램입니다. 랜덤한 종류의 차가 주유소로 들어와 특정 연료 종류와 양을 요구합니다. 올바르게 주유하여 주유소의 평판을 높이고, 돈을 일정량 이상 모아 승리하는 것이 최종 목표입니다.

[목적]

- 클래스 정의 및 인스턴스 생성을 익힙니다.
- 클래스 상속 및 메서드 오버라이딩을 익힙니다.

[주의사항]

- (1) 소스코드 저장 시 이름은 `assn3.py` 로 작성합니다.
- (2) 보고서는 `assn3.docx`, `assn3.hwp` 또는 `assn3.pdf` 로 저장 합니다.
- (3) 문서에 클래스 상속 구조 및 각 클래스에서 정의하여야 할 변수와 메서드가 설명되어 있으니 확인 후 구현하도록 합니다.
- (4) 명시된 예외처리 외에는 고려하지 않아도 됩니다.

[설명 및 요구사항]

1. 프로그램 초기 화면

프로그램을 처음으로 실행하면 그림 1과 같이 5개의 선택사항이 있는 메뉴를 출력하고, 사용자로 부터 5가지 선택사항 중 하나를 입력 받도록 합니다. 이때 올바른 입력은 0에서 4 사이의 정수이고, 이외에 다른 값 입력 시 "Wrong input!" 문구와 함께 다시 입력 받습니다. (예시의 빨간색 밑줄은 유저 입력에 해당합니다.)

```
-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select: 5

Wrong input!
Select:
```

그림 1. 프로그램 실행 첫 화면

2. 주유소 관리 기능

1) 현재 상태 확인 (2. Show current status)

그림 2 과 같은 초기 선택 메뉴 화면에서 2를 선택하면 아래의 예시처럼, 총 영업일, 평판, 보유 금액, 오늘 방문한 차량 수, 남은 연료의 양을 확인할 수 있습니다. 방문한 차량 수에는 돌려보낸 차량의 수도 포함됩니다.

```
-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select: 2

-----STATUS-----
Day 1
Rating: 0
Money: $1000.00
# Customers handled for today: 0
Diesel left: 100 Liters
Gasoline left: 100 Liters

-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select:
```

총 영업일 →
평판 →
보유금액 →
오늘 방문한 차량 수 →
남은 연료의 양 {

그림 2. 프로그램 시작 직후 현재 상태 확인

위 예시는 프로그램 시작 후, 첫 번째 선택으로 2를 입력한 예시로 주유소의 초기 상태를 확인할 수 있습니다. 프로그램 시작 시, 총 영업일은 1, 평판은 0, 보유금액은 1000.00달러, 오늘 방문한 차량 수는 0, Diesel 과 Gasoline 의 보유량은 각각 100리터에서 시작함을 확인할 수 있습니다.

아래 그림 3은 게임이 얼마간 진행된 뒤 총 영업일, 보유 금액, 오늘 방문한 차량 수, 평판, Diesel 과 Gasoline 의 보유량이 변경된 상태에서 현재 상태를 확인하는 경우입니다.

```
-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select: 2

-----STATUS-----
Day 5
Rating: 8
Money: $2420.79
```

```
# Customers handled for today: 2
Diesel left: 97 Liters
Gasoline left: 128 Liters

-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select:
```

그림 3. 게임 진행 도중 현재 상태 확인

그림 2 및 그림 3에서와 같이, 현재 주유소 상태가 출력 된 후, 초기 메뉴 화면이 다시 출력되어 사용자 입력을 기다립니다.

2) 주유소 탱크 채우기 (1. Refill tanks)

차량이 방문하여 주유를 하게 되면 탱크가 점차 바닥나므로, 상점을 이용해 탱크를 채워야 합니다. 초기 선택 메뉴 화면에서 1을 선택하면 탱크를 채울 수 있습니다. 상점이 보유하고 있는 Diesel 과 Gasoline 의 양은 무한대라고 가정합니다.

```
-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select: 1

Which one do you want to refill?
0. Diesel
1. Gasoline
Select: 0
Based on your rating 0, the discount ratio is 0.00%
The base unit buying price of Diesel for today is $9.00,
so the discount unit buying price will be $9.00

You have $1000.00. Amount of diesels to buy (liters): 4000
You don't have enough money

You have $1000.00. Amount of diesels to buy (liters): 100
Money spent: $1000 -> $100
Diesel refilled: 100 Liters -> 200 Liters

-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select:
```

그림 4. 주유소 탱크 채우기

초기에, selling price(운전자에게 판매)는 Gasoline: \$15 / L, Diesel: \$10 / L 입니다. buying price(상점에서 구매)는 항상 selling price 에서 10% 할인된 가격입니다.

위 그림 4 에서와 같이, 어떤 종류의 연료 탱크를 충전할 것인지 묻고, 주유소 평판(rating)에 따른 리터당 가격을 알려줍니다. 이때, 평판이 높을수록 할인율이 높아집니다. 충전할 양을 입력하였을 때, 지불해야 하는 금액이 현재 주유소가 보유한 금액보다 큰 경우는 구매가 불가능하다고 알려줍니다. 지불 가능한 경우 연료의 양 및 보유 금액을 업데이트합니다.

충전할 양 입력 시, 양수 이외의 값을 입력하면 탱크 충전을 취소 하고, 초기 메뉴 화면으로 돌아갑니다.

평판이 r이라고 할 때, 할인율은 $\min(\max(0, r/2), 30)$ 으로 정의됩니다. 예를 들어, $r = 10$ 인 경우 할인율은 $\min(5, 30) = 5$, $r = 80$ 인 경우 할인율은 $\min(40, 30) = 30$ 입니다. 아래 그림 5를 통해 평판에 따라 할인율 및 buying price가 결정되는 것을 확인할 수 있습니다.

```
Which one do you want to refill?
0. Diesel
1. Gasoline
Select: 1
Based on your rating 22, the discount ratio is 11.00%
The base unit buying price of Gasoline for today is
$13.35, so the discount unit buying price will be $11.88

You have $4891.27. Amount of gasolines to buy (liters):
```

그림 5. 평판에 따른 할인율 및 buying price

3) 차량 대기(0. Wait for a vehicle)

초기 선택 메뉴 화면에서 0을 선택하면 그림 6와 같이 차량 대기 모드로 들어갑니다. 이 경우, 차량이 주유하러 주유소를 방문할 때까지 기다리게 됩니다. 약 1초의 시간이 지난 후에 SUV, Hybrid, Bus, Truck 중의 1대가 방문합니다. (13쪽의 [Waiting...] 부분 참고)

```
-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select: 0
Waiting...

<<Vehicle Info>>
Fuel type: Diesel, Vehicle type: Truck, Fuel: 114 / 300
Driver: I'd like 110 liters, please.
Diesel left: 100 Liters, Gasoline left: 100 Liters

0. Change fueling method
1. Start fueling
2. Let go
Select:
```

그림 6. 차량 대기 모드

차량 대기 모드에서는 차량이 도착하기 전까지 ‘Waiting...’ 이 출력되며, 차량 도착 시 <<Vehicle Info>> 표시와 함께 차량의 연료 종류(Gasoline 혹은 Diesel), 연료통 크기, 차종(SUV, Hybrid, Bus, Truck), 남은 연료가 표시됩니다. 이때, 연료 종류 및 연료통 크기는 **그림 7** 와 같이 차종별로 정해져 있습니다.

| 차종 | 연료 종류 | 연료통 크기(L) |
|--------|----------|-----------|
| SUV | Gasoline | 80 |
| Hybrid | | 60 |
| Bus | Diesel | 100 |
| Truck | | 300 |

그림 7. 차종별 연료 종류 및 연료통 크기

운전자는 얼마만큼의 연료를 채우기 원하는지 알려주며, 이때 **그림 6** 에서와 같이 정확한 양뿐만 아니라 **그림 8** 에서처럼 가득(full)을 요구하기도 합니다. 가득 채우기를 요구받았을 때 올바르게 주유하는 방법은 두 가지로, 주유 방식을 full로 변경하는 것(탱크에 충분한 양의 연료가 있어야 합니다), 혹은 (연료통의 크기 - 남은 연료)만큼 직접 값을 입력해 주유하는 방법입니다.

3-1) 주유 방식 변경을 참조하세요.

그림 6 에서 도착한 차량은 연료가 Diesel, 차종이 Truck, 연료통의 크기는 300 리터, 남은 연료는 114 리터임을 알 수 있으며, 운전자는 110 리터를 주유하기를 요구하고 있습니다.

```
<<Vehicle Info>>
Fuel type: Gasoline, Vehicle type: Hybrid, Fuel: 8 / 60
Driver: please make it full!

0. Change fueling method
1. Start fueling
2. Let go
Select:
```

그림 8. 연료를 가득 채워 달라고 요구하는 경우

운전자가 주유를 요구할 때, 주유소에서는 선택할 수 있는 메뉴가 출력됩니다. **그림 8**의 예시처럼 0부터 2까지의 3개의 메뉴 중 하나를 선택할 수 있습니다.

3-1) 주유 방식 변경(Change fueling method)

차량이 도착해서 주유를 요구하면, 현재 주유 방식과 운전자에게 필요한 연료 종류 및 연료의 양이 맞는지 확인이 필요합니다. 그림 9 과 같은 차량이 도착한 상황에서 0을 선택(Change fueling method)하면 현재 주유 방식과 주유할 연료 양을 출력 후, 연료 종류 토글(Gasoline ↔ Diesel), 연료의 양을 변경할 수 있는 메뉴가 출력됩니다.

| | | |
|--|--|---|
| <p>차량도착</p> | <pre> <<Vehicle Info>> Fuel type: Gasoline, Vehicle type: Hybrid, Fuel: 8 / 60 Driver: please make it full! 0. Change fueling method 1. Start fueling 2. Let go Select: 0 Current Method: Diesel / 10 Liters. 0. Toggle fuel type 1. Change the amount of fuel 2. Finish Select: 0 Fuel type changed: Gasoline 0. Toggle fuel type 1. Change the amount of fuel 2. Finish Select: 1 Enter 'F' (full), or the Amount of liters to fuel: F Fueling method changed: Full 0. Toggle fuel type 1. Change the amount of fuel 2. Finish Select: 2 0. Change fueling method 1. Start fueling 2. Let go Select: 1 </pre> | |
| <p>주유 방식 변경 선택</p> | | |
| <p>연료 종류 변경 선택</p> | | <p>← 현재 선택된 주유 방식 출력 연료의 종류: 디젤 연료의 양: 10리터</p> |
| <p>연료 양 변경 선택</p> | | <p>← 연료 종류가 가솔린으로 변경됨</p> |
| <p>Full 선택</p> | | <p>← 연료 양이 Full 로 변경됨</p> |
| <p>연료 종류와 양 선택이 끝나면 Finish를 선택</p> | | |
| <p>1을 선택하여 주유 시작</p> | | |

그림 9. 주유 방식 변경

주유 방식은 초기에 가솔린 / 10 리터로 정해집니다. 현재 선택된 주유 방식과 도착한 차량의 요구사항에 맞춰 연료의 종류와 양을 변경합니다.

연료의 양 변경 시, 정확한 양 대신 'F'를 입력하면 차량의 연료통이 가득 찰 때까지 주유합니다. 원하는 대로 주유 방식을 변경 후 2 (Finish)를 입력하면 변경이 완료되고 이전 메뉴로 돌아갑니다. 연료 양을 직접 입력 시에는 양의 정수를 입력하여야 하며, 이외의 경우에는 "Wrong input!" 문구와 함께 다시 입력 받습니다.

3-2) 주유 시작 (Start fueling)

그림 9 에서 마지막 입력으로 1 (start fueling)을 선택한 경우, 올바른 주유 방식인지를 체크하여 결제가 이루어지고, 평판이 업데이트됩니다. 이후 차량은 주유소를 떠나고 초기 선택 메뉴가 출력되어 사용자 입력을 기다립니다.

가. 그림 9 과 같이 올바른 주유 방식이 선택된 경우에는 아래 그림 10 에서와 같은 상태 업데이트 메시지가 출력됩니다. 주유한 금액(주유한 양 * selling price)만큼 결제가 이루어지고, 운전자의 요구사항에 맞춰서 주유를 하여 평판(Rating)이 1 점 오릅니다.

```
Checking the conditions...
Money: $1000.00 -> $1780.00
Gasoline: 100 Liters -> 48 Liters

Driver: Thanks a lot!
Rating changed: 0 -> 1
```

그림 10. 올바른 주유

나. 연료 종류가 잘못된 경우, 아래 그림 11 에서와 같은 메시지가 출력되며 평판은 5 점 감소합니다. 실제 연료의 양이 줄지는 않으며, 평판만 나빠지고 차량은 주유소를 떠납니다.

```
Checking the conditions...
Requested: Diesel, Selected: Gasoline

System: This is not the right fuel type!
Rating: 0 -> -5
```

그림 11. 연료 종류 오류

다. 연료 종류는 올바르지만, 현재 주유소 연료 탱크에 있는 양보다 더 많은 연료를 사용하려 할 경우 아래 그림 12 에서와 같은 메시지가 출력되며, 평판이 1 점 감소합니다. 실제 연료의 양이 줄지는 않으며, 평판만 나빠지고 차량은 주유소를 떠납니다.

```
Checking the conditions...
Fuel type: Diesel
Amount of Diesel in the tank: 100 Liters, Tried: 120 Liters

System: There is not enough fuel in the tank!
Rating: 0 -> -1
```

그림 12. 연료 탱크 양 부족

라. 앞선 경우들을 제외하고, 현재 차량의 연료통에 채울 수 있는 양 보다 더 많은 양을 주유하려 할 경우 아래 그림 13 에서와 같은 메시지가 출력되며 평판은 3 점 감소합니다. 운전자는 연료통이 가득 찰 때까지 넣은 주유한 금액만큼 결제를 하고 주유소를 떠납니다.

```
Checking the conditions...
Fuel type: Gasoline
Maximum amount to fuel: 47 Liters, Tried: 50 Liters

Driver: Hey, it overflows! Stop there!
Money: $1000.00 -> $1705.00
Gasoline: 100 Liters -> 53 Liters
Rating: 0 -> -3
```

그림 13. 연료 과다 주입

마. 앞선 경우들을 제외하고, 연료 종류는 맞으나 요구한 양과 다른 양을 주유할 경우 아래 **그림 14** 에서와 같은 메시지가 출력되며 평판은 1 점 감소합니다. 운전자는 주유한 금액만큼 결제를 하고 주유소를 떠납니다.

```
Checking the conditions...
Fuel type: Gasoline
Requested: 20 Liters, Tried: 10 Liters

Driver: Well, not the exact amount, but thanks anyway!
Money: $1000.00 -> $1150.00
Gasoline: 100 Liters -> 90 Liters
Rating: 0 -> -1
```

그림 14. 연료 양 오류

3-3) 차량 돌려보내기 (Let go)

운전자가 요구한 기름의 양보다 현재 주유소 탱크에 있는 기름의 양이 부족하다고 판단하는 경우, 혹은 운전자의 서비스 요청(‘ 4) 운전자의 서비스 요청 ’ 참조)을 수락하고 싶지 않은 경우, 차량을 돌려보낼 수 있습니다. 이 경우 아래 **그림 15** 에서와 같은 메시지가 출력되며 평판은 1 점 감소합니다. 이후 차량은 주유소를 떠나고 초기 선택 메뉴가 출력되어 사용자 입력을 기다립니다.

```
<<Vehicle Info>>
Fuel type: Diesel, Vehicle type: Truck, Fuel: 78 / 300
Driver: please make it full!

0. Change fueling method
1. Start fueling
2. Let go
Select: 2

Owner: Currently, we are not available for that.
Driver: Well, see you then!
Rating changed: -3 -> -4
```

그림 15. 차량 돌려보내기

4) 운전자의 서비스 요청

때때로, 운전자가 서비스를 요청하는 경우가 있습니다. 각 차종별 요청할 수 있는 서비스는 **그림 16** 과 같습니다. 운전자에게 무료로 제공되며, 비용은 주유소가 지불하게 됩니다.

| 차종 | 서비스 | 비용 |
|----------------|-----------|-------|
| 디젤 자동차(버스, 트럭) | 요소수(DEF) | \$50 |
| SUV | 엔진오일 교체 | \$100 |
| Hybrid | 타이어 펑크 수리 | \$300 |

그림 16 차종별 서비스 요청

요청을 수락하는 경우 보유 금액은 줄어들지만 평판이 3 점 상승하며, 수락하지 않는 경우 평판이 1 점 감소합니다. 평판 업데이트 후, 차량은 주유소를 떠나고 초기 선택 메뉴가 출력되고 사용자 입력을 기다립니다.

아래 그림 17은 Truck의 서비스 요청을 수락한 예시입니다.

```
<<Vehicle Info>>
Fuel type: Diesel, Vehicle type: Truck, Fuel: 49 / 300
Driver: Refill the DEF, please.

Provide some DEF for free? (costs $50 yet increases rating by 3)
0. Yes
1. No
Select: 0

You provided some DEF for free!
Money: $1000 -> $950
Driver: I'm blessed to have all of these. Thanks!
Rating: 0 -> 3
```

그림 17. 서비스 요청 - 수락

아래 그림 18은 SUV의 서비스 요청을 거절한 예시입니다.

```
<<Vehicle Info>>
Fuel type: Gasoline, Vehicle type: SUV, Fuel: 22 / 80
Driver: An oil change, please.

Change the engine oil for free? (costs $100 yet increases rating by 3)
0. Yes
1. No
Select: 1

Owner: Currently, we are not available for that.
Driver: Well, see you then!
Rating changed: 0 -> -1
```

그림 18. 서비스 요청 - 거절

5) 다음 날로 가기 (3. Go to next day)

초기 선택 메뉴 화면에서 3을 선택하면 오늘의 영업을 종료하고 다음 날로 이동할 수 있습니다. 단, 3명 이상의 고객을 응대한 경우에만 가능합니다. 3명 미만인 경우 아래의 그림 19에서와 같이 오류 메시지 출력 후, 초기 선택 메뉴가 출력되어 사용자 입력을 기다립니다.

```
-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select: 3

You have to handle at least three customers. (2 / 3)
```

그림 19. 다음 날로 이동 - 실패

3명 이상을 응대한 경우, 그림 19에서와 같이 오늘의 영업일(Day 1) 및 기름 가격 변화를 보여주고, 다음 날로 넘어갑니다. 이때, 총 영업일은 1이 증가하고, 오늘 응대한 고객의 수는 0으로

초기화됩니다. 가솔린, 디젤 각각의 가격은 아래와 같이 랜덤으로 증감합니다(random 모듈 import 필요).


```
price[ 'Gasoline' ] *= (1 + random.uniform(-0.1, 0.1))  
price[ 'Diesel' ] *= (1 + random.uniform(-0.1, 0.1))
```

```
-----GAS STATION-----  
0. Wait for a vehicle  
1. Refill tanks  
2. Show current status  
3. Go to the next day  
4. End Game  
Select: 3  
  
Day 1 finished.  
Gasoline unit selling price: $15.00 -> $15.29  
Diesel unit selling price: $10.00 -> $10.87
```

그림 20. 다음 날로 이동 - 성공

6) 게임 종료 (4. End Game)

초기 선택 메뉴 화면에서 4를 선택하면 프로그램을 종료할 수 있습니다. 단, 보유 금액이 \$5000 이상일 때만 가능하며, \$5000 미만일 때는 아래의 오류 메시지를 아래 **그림 20**에서와 같이 출력 후 초기 선택 메뉴를 다시 출력하여 사용자 입력을 기다립니다.



```
-----GAS STATION-----
0. Wait for a vehicle
1. Refill tanks
2. Show current status
3. Go to the next day
4. End Game
Select: 4

You should have at least $5000 to finish the game.
You have: $3000.00
```

그림 21. 게임 종료 - 실패

정상적으로 종료될 때는 아래 **그림 21**에서와 같이, 평판, 보유 금액, 현재까지 총 고객 수를 포함한 ‘요약’을 출력합니다. 마지막으로 ‘You win!’이 출력되며 프로그램이 종료됩니다.

```
-----Summary-----
Rating: 27
Money: $5420.79
Total customers handled: 42
-----
You win!
```

그림 22. 게임 종료 - 성공

[메인 파트 코드]

구현 시 다음의 코드를 활용하세요.

```
S = Station()
while True:
    if S.customer == None: # 차량이 도착하기 전(초기 상태 포함)
        if S.default_screen(): # 4. End Game 조건이 만족된 경우 True 를 반환하여 프로그램 종료
            break
    else: # 차량이 도착한 경우
        S.serve() # 차량의 요구에 따라 응대
```

[random 모듈 사용 부분]

- 새로 도착한 차량의 차종별 등장 확률은 SUV, Hybrid - 1/6, Bus, Truck - 1/3 입니다. 예를 들어, 차종을 결정하는 변수가 `car = random.randint(1, 6)`라고 하면, `car` 이 1 일 경우 SUV, 2 일 경우 Hybrid, 3 혹은 4 일 경우 Bus, 5 혹은 6 일 경우 Truck 이 생성되도록 할 수 있습니다.
- 도착한 차량이 주유를 요구할 확률은 차종에 관계없이 4/5, 서비스를 요구할 확률은 1/5 입니다. 문서 후반부 클래스 설명 부분에 등장하는 `class Station / def serve(self)` 함수 내에서 구현하는 것을 추천합니다.
- 차량이 주유 요구 시, 가득 채워달라고 요청(full)할 확률은 1/4 입니다. 이외의 경우, (연료통의 크기 - 남은 연료)의 약 50~80% 정도가 차도록 특정 양을 요구합니다. 문서 후반부의 클래스 설명 부분에서 `class Car / def cal_fuel(self)` 를 참조하세요.
- 차량의 남은 연료를 결정: 문서 후반부의 클래스 설명 부분에서 `class Car / def cal_fuel(self)` 를 참조하세요.

[random.seed(xxx)를 사용한 디버깅]

`random` 모듈이 사용되므로, 매 회 시행 결과가 달라 디버깅에 어려움을 겪을 수 있습니다. 아래와 같이 `random.seed(xxx)`를 사용하고 `xxx`에 동일한 수를 넣으면 항상 같은 결과를 얻게 됩니다.

```
import random
random.seed(213) # 과제 제출 시, 해당 코드는 주석 처리 후 제출합니다.
```

[Waiting...]

구현 시 다음의 코드를 활용하세요. `time.sleep(t)`은 `t` 초 동안 프로그램의 작동을 멈추는 함수입니다. 아래의 코드에서는 총 세 번, 각각 0.3 초 동안 프로그램이 멈추므로, 약 1 초 동안 차량을 기다리게 됩니다.

```
import time
print("Waiting...", end='')
for i in range(3):
    print(".", end='')
    time.sleep(0.3)
print()
```

[주유 결과 대화문 + 평판 업데이트]

‘3-2) 주유 시작’에서 출력되는 문구 및 평판 업데이트를 위해 아래의 딕셔너리를 사용하세요.

```
# rating_results
rs= {
    'for_free': [3, "Driver: I'm blessed to have all of these. Thanks!"],
    'let_go': [-1, "Owner: Currently, we are not available for that.\nDriver: Well, see you then!"],
    'good_job': [1, "Driver: Thanks a lot!"],
    'wrong_fuel': [-5, "System: This is not the right fuel type!"],
    'overflow': [-3, "Driver: Hey, it overflows! Stop there!"],
    'different': [-1, "Driver: Well, not the exact amount, but thanks anyway!"],
    'not_enough': [-1, "System: There is not enough fuel in the tank!"],
}
```

[숫자 자료형 출력 방식]

보유 금액을 나타내는 변수는 float 형으로 저장합니다. 출력 시에는 %.2f 를 사용해 소수점 둘째 자리까지만 표시합니다. 이외 (주유 시 / 상점 구매 시) 기름의 양, 날짜, 평판 등 나머지 모든 변수는 int 형으로 저장합니다.

[클래스 구현 시 요구 사항]

본 과제에서 사용되는 클래스의 상속 구조는 아래 그림 22 에서와 같습니다.

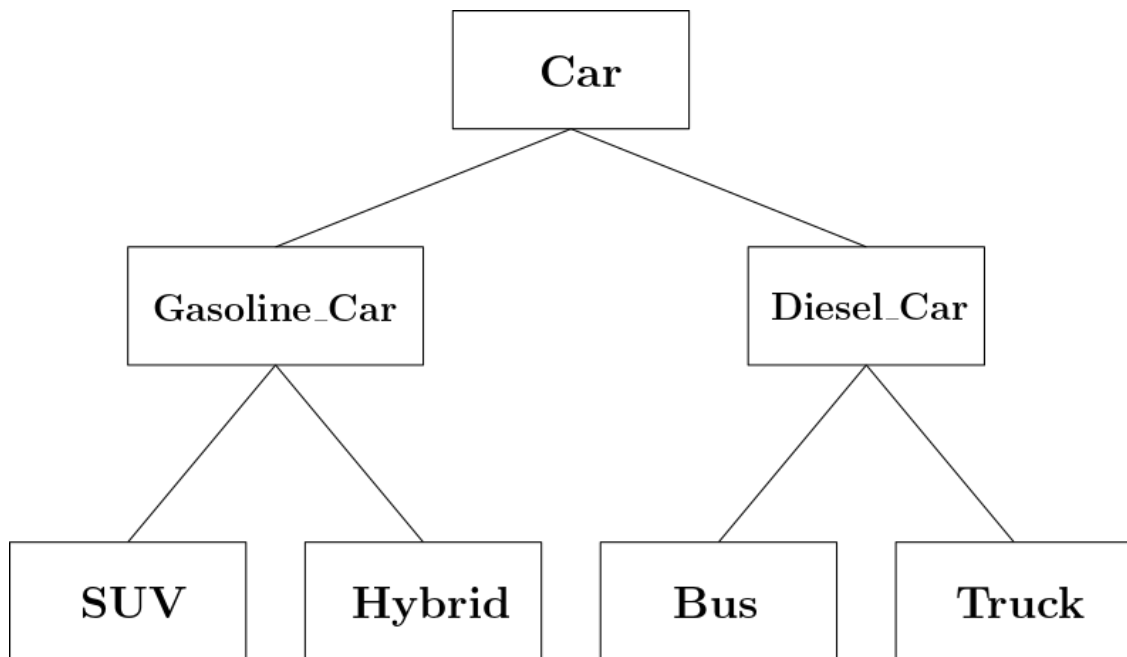


그림 23. 클래스 상속 구조

[클래스 변수 및 메서드]

다음 클래스 변수 및 메서드들을 반드시 작성하여 프로그램을 구현해야 합니다. 아래 명시된 변수 및 메서드 이름은 변경하지 말아주세요. 메서드의 매개변수의 개수 및 리턴 값 등은 자유롭게 변경 가능합니다. 그러나 동일한 기능을 하는 메서드는 반드시 있어야 하며, 변경 시 무엇을 어떻게 변경했는지 보고서에 기록하도록 합니다. 이 외에 필요한 메서드는 정의해서 사용할 수 있습니다.

class Station:

```
# 인스턴스 변수
self.day # 현재까지의 총 영업일(Day XX)을 표시하는 변수
self.rating # 주유소의 평판
self.money # 보유 금액
self.today_num # 오늘의 고객 수
self.total_num # 전체 고객 수
self.customer # 현재 응대 중인 고객. 없는 경우 None

# 메서드
def state_update(self): # 고객 응대에 따라 보유 금액, 평판 등을 업데이트하는 함수
def refill(self): # 상점으로부터 기름을 구매하는 함수
def print_status(self): # 현재 상태를 출력하는 함수
def default_screen(self): # 초기화면 출력 및 초기화면에서 (0. Wait for a vehicle)
    # 제외한 나머지 입력을 처리하는 함수
def serve(self): # 초기화면에서 (0. Wait for a vehicle) 선택 시, 실행되는 함수
    # 차량 도착 시 요구하는 서비스를 랜덤으로 결정하고, 이에 대한 응대
def price_update(self): # 기름 가격을 변동하는 함수
def next(self): # 다음 날로 이동하는 함수
```

class Car:

```
# 인스턴스 변수
self.fuel_type # 연료 종류(가솔린, 디젤)
self.vehicle_type # 자동차 종류(트럭, 버스, SUV, 하이브리드)
self.capacity # 연료통 크기
self.cur_fuel # 현재 남은 용량
self.needed # 주유 요구 시, 채워달라고 요청하는 연료의 양
self.full # 주유 요구 시, 연료통을 가득 채워달라고 하는지 여부

# 메서드
def __init__(self):
    self.cal_fuel()

def cal_fuel(self): # 남은 기름의 양 계산 및 full 여부
    self.cur_fuel = int(self.capacity * random.uniform(0.1, 0.4))
    if random.randint(1, 4) == 1:
        self.full = True
        self.needed = self.capacity - self.cur_fuel
    else:
        self.full = False
        self.needed = ((self.capacity - self.cur_fuel) *
random.uniform(0.5, 0.8) // 5) * 5

def printInfo(self): # 차량 정보 출력
    print("\n<<Vehicle Info>>")
    ... (채울 부분) ...
```

```

def fuel(self): # 지정한 주유 방식으로 차량에 기름을 넣는 함수
    print("Checking the conditions...")
    ... (채울 부분) ...

class Gasoline_Car(Car):
    # 메서드
    def __init__(self):
        super().__init__()
        self.fuel_type = "Gasoline"

class Diesel_Car(Car):
    # 메서드
    def __init__(self):
        super().__init__()
        self.fuel_type = "Diesel"

def upgrade_claim(self): # 업그레이드 요청
    print("Driver: Refill the DEF, please.")
    print("Provide some DEF for free? (costs $50 yet increases rating by 3)")

def upgrade(self) : # 업그레이드 요청에 응했을 때 실행되는 부분
    print("You provided some DEF for free")

class SUV(Gasoline_Car):
    # 메서드
    def __init__(self):
        ... (채울 부분) ...

def upgrade_claim(self): # 업그레이드 요청
    print("Driver: An oil change, please.")
    print("Change the engine oil for free? (costs $100 yet increases rating by 3)")

def upgrade(self) : # 업그레이드 요청에 응했을 때 실행되는 부분
    print("You replaced the engine oil for free")

class Hybrid(Gasoline_Car):
    # 메서드
    def __init__(self):
        ... (채울 부분) ...

def upgrade_claim(self): # 업그레이드 요청
    print("Driver: My car has flat tires...")
    print("Change the tires for free? (costs $300 yet increases rating by 3)")

def upgrade(self) : # 업그레이드 요청에 응했을 때 실행되는 부분
    print("You provided some tires for free")

```



```
class Bus(Diesel_Car):  
    # 메서드  
    def __init__(self):  
        ... (채울 부분) ...
```

```
class Truck(Diesel_Car):  
    # 메서드  
    def __init__(self):  
        ... (채울 부분) ...
```