CSC148 Worksheet 6 Solution

Hyungmo Gu

April 19, 2020

Question 1

- The two classes already defined are: Vehicle and SuperDuperManager

  The additional classes required to create for this exercise are: Car, Helicopter and MagicCarpet

  Car, Helicopter and MagicCarpet are child classes of Vehicle.

Question 2

a. The following are attributes possessed by all vehicles

- type
- initial_position
- moves_to
- move_diagonally
- fuel_usage

Correct Solution:

The following are attributes possessed by all vehicles

- position
- fuel

b. No. Referencing the following code in *worksheet_6_starter_code.py*,

```
class Vehicle:
    ...

    def __init__(self, initial_fuel: int,
                 initial_position: Tuple[int, int]) -> None:
        ...

        self.fuel = initial_fuel
        self.position = initial_position

```

we can come up with the following examples.

- Vehicle(100, (10,20))
- Vehicle(50, (5, 10))

Here, we can see the two vehicles have different value of fuel and initial position.

c. *fuel_needed* not implemented because each child classes have different fuel consumption rate, and the method is to be defined by the child classes by overriding it.

d. The following methods must be defined in each of its subclasses

- Car
  - fuel_needed
  - move
- Helicopter
  - fuel_needed
  - move
- MagicCarpet
  - __init__
  - move

Correct Solution:

- Car
  - __init__
    - Necessary because the parameter *position* must be set as optional
    - Necessary because *self.position* must default to (0,0) if the argument of position not given.
  - fuel_needed
    - Necessary because vehicle uses fuel

        * Necessary because needs to define the fuel cost based on it not being able to moving diagonally.

- Helicopter
  - __init__
    * Necessary because the parameter *position* must be set as optional
    * Necessary because *self.position* must default to (3,5) if the argument of position not given.
  - fuel_needed
    * Necessary because vehicle uses fuel
    * Necessary because needs to define the fuel cost based on it being able to move diagonally.

- MagicCarpet
  - __init__
    * Necessary to set the parameters *initial_fuel*, *initial_position* as optional
    * Necessary to randomize the value of *self.position*.
  - move
    * Necessary to set the parameters *new_x* and *new_y* as optional.
    * Necessary to randomize the value of new position.

Question 3

```
"""
Initializing SuperDuperManager:
>>> s = SuperDuperManager()
>>> s._vehicles
{}

Adding Vehicles:
>>> s.add_vehicle('Car', '1', 100)
>>> s._vehicles['1'].__class__.__name__
'Car'
>>> s.add_vehicle('Helicopter', '1', 100)
>>> s._vehicles['1'].__class__.__name__
'Car'

>>> s.add_vehicle('Helicopter', '2', 100)
>>> s._vehicles['2'].__class__.__name__
'Helicopter'

>>> s.add_vehicle('UnreliableMagicCarpet','3',100)
>>> s._vehicles['3'].__class__.__name__
'UnreliableMagicCarpet'
```

```
    Moving Vehicle:
    >>> s._vehicles['1'].position
    (0,0)
    >>> s.move_vehicle('1', 1, 1)
    >>> s._vehicles['1'].position
    (1,1)

    >>> s._vehicles['2'].position
    (3,5)
    >>> s.move_vehicle('2', 1, 1)
    >>> s._vehicles['2'].position
    (4,6)

    >>> s._vehicles['3'].position
    (4,8)
    >>> s._vehicles['3'].position
    (12,4)
    >>> s.move_vehicle('3', 1, 1)
    >>> s._vehicles['3'].position
    (100,100)

    Get Vehicle Position:
    >>> s.get_vehicle_position('1')
    (1,1)

    >>> s.get_vehicle_position('2')
    (4,6)

    >>> s.get_vehicle_position('3')
    (50,200)

    Get Vehicle Fuel:
    >>> s.get_vehicle_fuel('1')
    98

    >>> s.get_vehicle_fuel('2')
    99

    >>> s.get_vehicle_fuel('2')
    100
    """
```

Question 4

a. The instance attribute *id_* is used to keep track of vehicles.

The type of the instance attribute is string.

Correct Solution:

The instance attribute *self._vehicles* is used to keep track of vehicles.

The type of the instance attribute is 'dictionary'.

b. The vehicles are initialized in class *SuperDuperManager*'s *add_vehicle* method.

Correct Solution:

The instance attribute is initialized in class *SuperDuperManager*'s *__init__* method.

c. In code that keeps track of all the vehicles, the vehicles are updated via the methods *add_vehicle* and *move_vehicle*

Question 5

a. If left as is, every car object would possess the instance attributes *self.position* and *self.fuel* from class Vehicle.

b. No. Other instance attributes are not necessary.

Looking at the table provided at the beginning of worksheet, car class needs information about initial position, final position, fuel usage, and remaining fuel amount.

For initial and final position, the instance attribute *position* is used.

For fuel usage, this is embedded inside *fuel_needed* method. So, the additional attribute is not necessary.

For fuel remaining, the instance attribute *self.fuel* is used.

Notes:

- 여보, 형모 노래 듣고 있었어요
- https://www.youtube.com/watch?v=5jl8mzCaCr0
- 여보, 형모 내 여보 많이 보고싶어요
- 구래두 여보, 형모 내 여보 보러 꾹 참꾸 걸어요
- 여보, 사랑해
- 여보, 고마워요 :)

Question 6

a. Car inherits the following methods from class Vehicle: *__init__*, *fuel_needed* and *move*.

b. Of these inherited methods, *fuel_needed* needs to be implemented by overriding it.

c. The two methods needs overriding: *__init__* and *fuel_needed*.

For *__init__*, because we need to set the parameter *initial_position* as optional while keeping the rest of the code the same, we want to call the parent method as a helper

```
    class Car:

        def __init__(self, initial_fuel: int,
                     initial_position: Tuple[int, int] = (0,0)) -> None:

            super().__init__()


```

For *fuel_needed*, because we need to replace contents within, the parent class method shouldn't be called as a helper.

```
    class Car:
        def fuel_needed(self, new_x: int, new_y: int) -> int:
            # New lines of code here


```

Notes:

- 여보!!!!!!!!
- 사랑해요 여보
- 형모 당신만을 사랑해

Question 7

Code is also included in *worksheet_6_solution.py*.

```
    from worksheet_6_starter_code import Vehicle, SuperDuperManager

    """
    Question 7
    """

    class Car(Vehicle):

        def __init__(self, initial_fuel: int,
                     initial_position: Tuple[int, int] = (0,0)) -> None
    :
```

```
            super().__init__()

        def fuel_needed(self, new_x: int, new_y: int) -> int:

            old_x = self.position[0]
            old_y = self.position[1]

            delta_x = abs(old_x - new_x)
            delta_y = abs(old_y - new_y)

            return delta_x + delta_y
```

Question 8

1) Helicopter

- **Question 5:**

  a. If left as is, every helicopter objects would possess instance attributes *self.position* and *self.fuel* from class Vehicle.

  This is the same as Car class.

  b. No. With the same reasoning as Car class, *self.position* and *self.fuel* are sufficient to do all required operations.

- **Question 6:**

  a. Just like Car class, Helicopter Class inherits the following methods from class Vehicle: *__init__*, *fuel_needed* and *move*.

  b. Just like Car class, *fuel_needed* needs to be implemented by overriding it.

  c. With the same reasoning as Car class, the two methods needs overriding: *__init__* and *fuel_needed*.

```
from worksheet_6_starter_code import Vehicle, SuperDuperManager
import math

"""
Question 8.1
"""

class Helicopter(Vehicle):

    def __init__(self, initial_fuel: int,
                 initial_position: Tuple[int, int] = (0,0)) -> None
:
        super().__init__()

    def fuel_needed(self, new_x: int, new_y: int) -> int:
```

```
15
16              old_x = self.position[0]
17              old_y = self.position[1]
18
19              delta_x = abs(old_x - new_x)
20              delta_y = abs(old_y - new_y)
21
22              return math.ceil(math.sqrt(delta_x**2 + delta_y**2))
```

2) UnreliableMagicCarpet

- **Question 5:**

  a. If left as is, every UnreliableMagicCarpet object would possess the instance attributes *self.position* and *self.fuel* from class Vehicle.

  b. No. With the same reasoning as Car class, other instance attributes are not necessary.

- **Question 6:**

  a. UnreliableMagicCarpet inherits the following methods from class Vehicle: *__init__*, *fuel_needed* and *move*.

  b. No inherited methods must be implemented. All methods are implemented by parent class.

  Correct Solution

  fuel_needed needs implementation. This is to suppress 'NotImplemented' error.

  c. The two methods needs overriding: *__init__*, and *move*.

  For *__init__*, the following needs modification

  - *initial_position*: This argument needs to be set as optional
  - *self.position*: This instance attribute needs to be set a random integer value

  Because we want to change the above while keeping information about *self.fuel*, we want to use parent class method as a helper.

```
1         import random
2         from datetime import datetime
3
4         class UnreliableMagicCarpet:
5
6             def __init__(self, initial_fuel: int,
7                          initial_position: Tuple[int, int] = (0,0)
  ) -> None:
```

```
                super().__init__()

                random.seed(datetime.now())
                self.position = (random.randint(-100,100), random.
randint(-100,100))


```

Now, for *move*, we need to replace entire contents within.

Then, it follows from this fact that parent method doesn't need to be called as a helper.

```
        class UnreliableMagicCarpet:
            def move(self, new_x: int, new_y: int) -> None:
                # New lines of code here


```

Correct Solution

The two methods needs overriding: *__init__*, and *move*.

For *__init__*, the following needs modification

- *initial_position*: This argument needs to be set as optional
- *self.position*: This instance attribute needs to be set a random integer value

Because we want to change the above while keeping information about *self.fuel*, we want to use parent class method as a helper.

```
            import random
            from datetime import datetime

            class UnreliableMagicCarpet:

                def __init__(self, initial_fuel: int,
                             initial_position: Tuple[int, int]
= (0,0)) -> None:
                    super().__init__()

                    random.seed(datetime.now())
                    self.position = (random.randint(-100,100),
random.randint(-100,100))


```

Now, for *fuel_needed*, we need to replace entire contents within.

It follows from this fact that parent method doesn't need to be called as a helper.

```
            class UnreliableMagicCarpet:
                def fuel_needed(self, new_x: int, new_y: int)
    -> None:
                    # New lines of code here


```

Now, for *move*, we need to replace entire contents within.

Then, it follows from this fact that parent method doesn't need to be called as a helper.

```
            class UnreliableMagicCarpet:
                def move(self, new_x: int, new_y: int) -> None
    :
                    # New lines of code here


```

```
    from worksheet_6_starter_code import Vehicle, SuperDuperManager
    import math

    """
    Question 8.2
    """

    class UnreliableMagicCarpet:

        def __init__(self, initial_fuel: int,
                     initial_position: Tuple[int, int] = (0,0)) -> None
    :
            super().__init__()

            random.seed(datetime.now())
            self.position = (random.randint(-100,100), random.randint
    (-100,100))

        def move(self, new_x: int, new_y: int) -> None:
            self.position = (random.randint(-100,100), random.randint
    (-100,100))
```

Correct Solution

```

    from worksheet_6_starter_code import Vehicle, SuperDuperManager
```

```
    import math

    """
    Question 8.2
    """

    class UnreliableMagicCarpet:

        def __init__(self, initial_fuel: int,
                     initial_position: Tuple[int, int] = (0,0)) ->
    None:
            super().__init__()

            random.seed(datetime.now())
            self.position = (random.randint(-10,10), random.randint
    (-10,10)) #<- Correct solution

        def fuel_needed(self, new_x: int, new_y: int) -> int:
            return 0 # <- Correct Solution

        def move(self, new_x: int, new_y: int) -> None:
            new_x = self.position[0] + random.randint(-2, 2) # <-
    Correct Solution
            new_y = self.position[1] + random.randint(-2, 2) # <-
    Correct Solution
            self.position = (new_x, new_y) # <- Correct Solution


```