CSC148 Worksheet 5 Solution

Hyungmo Gu

April 17, 2020

Question 1

```
"""
from datetime import date, timedelta

...

misbehaved_1 = Tweet('', date.today(), 'test')
misbehaved.like(-1)

yesterday = date.today() - timedelta(days=1)
misbehaved_2 = Tweet('john', yesterday 'test')

misbehaved_3 = Tweet('john', date.today(), '')
"""
```

Correct Solution:

```
        """
        from datetime import date, timedelta

        ...

        misbehaved_1 = Tweet('', date.today(), 'test')
        misbehaved.like(-1)

        yesterday = date.today() - timedelta(days=1)
        misbehaved_2 = Tweet('john', yesterday 'test')

        misbehaved_3 = Tweet('john', date.today(), '')

        misbehaved_4 = Tweet('john doe', date.today(), 'test') # More
    solution!!
        """

```

Question 2

- A tweet must receive positive number of likes
- A tweet must have user id without spaces

Correct Solution:

- A tweet must receive non-negative number of likes
- A tweet user id must not be empty.
- A tweet must have user id without spaces
- A tweet must have content length less than or equal to 280 characters

Question 3

- Fix for the property 'a tweet must receive positive number of likes'

```
    class Tweet:
        ...

        def like(self, n: int) -> None:
            """Record the fact that this tweet received <n> likes.
            These likes are in addition to the ones <self> already has
    .
            """

            if n < 0:
                raise ValueError('n must be non-negative value')

            self.likes += n

```

Correct Solution:

```
            class Tweet:
            ...

            def like(self, n: int) -> None:
                """Record the fact that this tweet received <n>
    likes.
                These likes are in addition to the ones <self>
    already has.

                Precondition: n >= 0 # Correct Solution
```

```
                """

                self.likes += n

```

- Fix for the property 'a tweet user id must not be empty.'

```
    class Tweet:
        ...

        def __init__(self, who: str, when: date, what: str) -> None:
            """Initialize a new Tweet.
            """

            if who.strip() == '':
                raise ValueError("variable 'who' must not be empty")

            self.userid = who
            self.content = what
            self.created_at = when
            self.likes = 0

```

Correct Solution:

```
        class Tweet:
            ...

            def __init__(self, who: str, when: date, what: str) ->
    None:
                """Initialize a new Tweet.

                Precondition: who.strip() != '' # Correct Solution
                """

                self.userid = who
                self.content = what
                self.created_at = when
                self.likes = 0

```

- Fix for the property 'a tweet must have user id without spaces'

```
    class Tweet:
        ...

        def __init__(self, who: str, when: date, what: str) -> None:
            """Initialize a new Tweet.
            """

            if ' ' in who:
```

```
                raise ValueError("variable 'who' must not have spaces"
    )

            self.userid = who
            self.content = what
            self.created_at = when
            self.likes = 0

```

Correct Solution:

```
        class Tweet:
            ...

            def __init__(self, who: str, when: date, what: str) ->
    None:
                """Initialize a new Tweet.

                Precondition: ' ' not in who # Correct Solution
                """

                self.userid = who
                self.content = what
                self.created_at = when
                self.likes = 0

```

- Fix for the property 'a tweet must have content length less than or equal to 280 characters'

```
    class Tweet:
        ...

        def __init__(self, who: str, when: date, what: str) -> None:
            """Initialize a new Tweet.
            """

            ...

            if len(what) > 280:
                raise ValueError("variable 'what' must be 280
    characters or less")

            self.userid = who
            self.content = what
            self.created_at = when
            self.likes = 0

```

Correct Solution:

```
        class Tweet:
            ...

            def __init__(self, who: str, when: date, what: str) ->
    None:
                """Initialize a new Tweet.

                Precondition: len(what) <= 280 # Correct Solution
                """

                self.userid = who
                self.content = what
                self.created_at = when
                self.likes = 0

```

Notes:

- Learned that 'Precondition: len(what) <= 280' is called **representational invariant**.
- Learned that **representational invariant** is a property of instance attribute that every class must satisfy.

Question 4

a. It is insufficient to implement the *best_percentage*. Consider the following example

1. One or more elements in <teams>with empty strings (i.e. ['', '', ''])
2. One or more elements in <teams>with same name (i.e. ['a', 'a', 'b'])
3. <team1>or <team2>as empty string (i.e. team1 = '', team2 = '')
4. <team1>or <team2>containing the same value (i.e. team1 = 'a', team2 = 'a')
5. <score1>or <score2>as negative value (i.e. score1 = -1, score2 = -4)

Attempt 2:

It is insufficient to implement the *best_percentage*. Consider the following example

1. One or more elements in <teams>with empty strings (i.e. [‘’, ‘’, ‘’])
   - In this case, only one team will be added to dictionary. This would result in <team1>and <team2>being the same value. This results in the misbehavior of calculating tournament score based on the same team.
2. All elements in <teams>with same name (i.e. [‘a’, ‘a’, ‘a’])
   - In this case, this leads to the same misbehavior as case 1.
3. <team1>or <team2>as empty string or as same value (i.e.1 team1 = ‘’, team2 = ‘’), (i.e.2 team1 = ‘a’, team2 = ‘a’)
   - In this case, the instance attribute leads to undesired behavior of calculating tournament score based on the same team.

b. One example is *t.record_game(‘a’,‘b’,0,0)*.

The following precondition should be added such that the input attributes support the statistics.

```
1    def record_game(self, team1, team2, score1, score2):
2        """
3        ...
4
5        Precondition: team1 and team2 are both in this tournament.
6        Precondition: <score1> != <score2>
7        """
8
9
```

c. When instance attributes are changed, relevant pre-conditions and post-conditions become invalidated, and this would result in one or more methods to misbehave.