Railway station

Project 6

Савинов Владислав, 19Б03-мкн

Формат входных данных

Входные данные

- Количество поездов, событий и “связей”
- Дата
- Связи
- Описание поезда: train_id, amount of carriages, type
- Событие: train_id, event

Events

- add [amount] [time]
- remove [amount] [time]
- schedule [arrival] [departure]
- delay arrival [minutes] [time]
- delay departure [minutes] [time]

```
USAGE = '''USAGE:
Schedule input:

MARKS:
∗ N —— amount of trains that arrives at the platform during the day
∗ M —— amount of events
∗ K —— amount of connections

FORMAT:
1st line: [day/month/year] —— date of the schedule
2nd line: [N] [M] [K] —— amount of trains, events and connections
3rd line: [amount of platforms] [amount of ways]
K lines: [connection]
N lines: [train number] [amount of carriages] [train type]
M lines: [train number] [event]

CONNECTION DESCRIPTION (if platform and way are connected):
[platform number] [way number]

TRAIN TYPES:
P    :     passenger
F    :     freight
E    :     train with the destination / origin = this platform

POSSIBLE EVENTS:
1. add [amount] [time]    :    add [amount] of carriages in [time]
2. remove [amount] [time]    :    remove [amount] of carriages in [time]
3. schedule [arrival] [departure]    :    set the arrival & departure time
format    :    day/month/year hours:minutes
4. delay arrival [minutes] [time]    :    delay the arrival for [minutes] minutes in
[time]
5. delay departure [minutes] [time]    :    delay the departure for [minutes] minutes
in [time]
'''
```

Данные

```python
from datetime import datetime, timedelta
from constants import MINUTES_DAY, MINUTES_HOUR

ways2platforms = {}
schedule = {}
availability = {}
trains = {}
events = []

def initialize(date):
        date_init = datetime.strptime(date, ‘%d/%m/%Y').date() -\
                                         timedelta(days=1)
        time_init = datetime.strptime('23:30', ‘%H:%M').time()
        datetime_start = datetime.combine(date_init, time_init)

        for delta in range(0, MINUTES_DAY + 30):
                time_value = datetime_start + \
                        timedelta(days=delta // MINUTES_DAY,
                                 hours=(delta % MINUTES_DAY) // \
                                     MINUTES_HOUR
                                 minutes=delta % MINUTES_HOUR)
                availability[time_value] = set()
                schedule[time_value] = []
```

Абстрактный поезд

```python
from abc import ABC, abstractclassmethod
from datetime import datetime

class AbstractTrain(ABC):
    def __init__(self,
                train_id: str = None,
                carriage_num: int = None,
                platform: int = None,
                way: int = None,
                arrival: str = None,
                departure: str = None):

        self.__train_id = train_id
        self.__carriage_num = carriage_num
        self.platform = platform
        self.way = way
        if not arrival:
            self.__arrival = None
        else:
            self.__arrival = datetime.strptime(arrival,
                                               ‘%d/%m/%Y %H:%M')
        if not departure:
            self.__departure = None
        else:
            self.__departure = datetime.strptime(departure,
                                                 '%d/%m/%Y %H:%M’)
```

```python
@property
def train_id(self):
    return self.__train_id

@property
def arrival(self):
    return self.__arrival

@property
def departure(self):
    return self.__departure

@property
def carriage_num(self):
    return self.__carriage_num
```

```python
from data import schedule, availability
from datetime import datetime
from messages import LESS_CAR

def remove_carriages(self, amount, time_value):
        '''
        Removing [amount] of carriages in time
        '''
        time_value = datetime.strptime(time_value, '%d/%m/%Y %H:%M')
        if self.carriage_num >= amount:
            self.__carriage_num = self.carriage_num - amount
        else:
            self.__carriage_num = 0
        schedule[time_value].append(LESS_CARR.format(amount,
                                                     self.train_id,
                                                     time_value))

def set_arrival(self, arrival_time):
        '''
        Setting the arrival time as [arrival]
        '''
        try:
            arrival_input = datetime.strptime(arrival_time,
                                              ‘%d/%m/%Y %H:%M’)
            self.__arrival = arrival_input
        except:
            raise TypeError('Incorrect arrival format!’)
```

```python
@abstractclassmethod
    def set_platform(self, setting_time):
        '''
        Setting the platform randomly at [setting_time]
        '''
        pass

@abstractclassmethod
    def arrive(self):
        '''
        Accepting the train
        '''
        pass

@abstractclassmethod
    def process_train(self):
        '''
        Running all the required functions to process train
        '''
        pass
```

```
from data import availability
from datetime import datetime

def set_way(self, setting_time):
       '''
       Setting the way randomly at [setting_time]
       '''
       setting_time = datetime.strptime(setting_time,
                                        ‘%d/%m/%Y %H:%M')
       self.way = random.choice(list(availability[setting_time]))

       for delta in range(0, (self.departure - \
                              self.arrival).seconds // 60):
           time_value = setting_time + \
                        timedelta(days=0,
                                  hours=delta // MINUTES_HOUR,
                                  minutes=delta % MINUTES_HOUR)
           try:
               availability[time_value].remove(self.way)
           except:
               raise Exception(’The train crashed..’)
```

Пассажирский поезд

passenger_train.py

```python
import random
from data import schedule, ways2platforms
from messages import PASSENGER_PLATFORM
from train import AbstractTrain

class PassengerTrain(AbstractTrain):
            …

def set_platform(self):
        self.platform = random.choice(ways2platforms[self.way])
        delta = timedelta(minutes=10)
        setting_time = self.arrival - delta
        schedule[setting_time].append(PASSENGER_PLATFORM.format(
                                      self.train_id,
                                      self.platform,
                                      self.arrival))

def process_train(self):
        self.arrive()
        self.set_way()
        self.set_platform()
        self.depart()
```

```
def delay_arrival(self, delay_time, changes_time):
       super().delay_arrival(delay_time,
                             changes_time,
                             PASSENGER_DELAY_ARRIVAL)

def delay_departure(self, delay_time, changes_time):
       super().delay_departure(delay_time,
                               changes_time,
                               PASSENGER_DELAY_DEPARTURE)
```

Сообщения

messages.py

```
PASSENGER_ARRIVAL = "The passenger train №{} is currently arriving"

PASSENGER_PLATFORM = "The passenger train №{} will arrive at the platform №{} at {}"

PASSENGER_WAY = "For the passenger train №{} the track №{} was chosen"

PASSENGER_DELAY_ARRIVAL = "Arrival of the passenger train №{} is delayed for {} minutes"

PASSENGER_DELAY_DEPARTURE = "Departure of the passenger train №{} is delayed for {} minutes"

PASSENGER_DEPARTURE = "The passenger train №{} is currently departing"


FREIGHT_ARRIVAL = "The freight train №{} is currently arriving"

FREIGHT_PLATFORM = "The freight train №{} will arrive at the platform №{} at {}"

FREIGHT_WAY = "For the freight train №{} the track №{} was chosen"

FREIGHT_DELAY_ARRIVAL = "Arrival of the freight train №{} is delayed for {} minutes"


FORMED_ARRIVAL = "The train №{} is ready for the boarding"

FORMED_PLATFORM = "The train №{} will be formed at the platform №{}"

FORMED_WAY = "The train №{} will be formed near the track №{}"

FORMED_DELAY_ARRIVAL = "The train №{} is not ready yet, readiness is delayed for {} minutes"

FORMED_DELAY_DEPARTURE = "The train's №{} departure is delayed for {} minutes"

FORMED_DEPARTURE = "The train №{} is currently departing"

NEW_CARR = "{} carriages were added to the train №{} at {}"

LESS_CARR = "{} carriages were removed from the train №{} at {}"

SEPARATOR = "########################################"
```

Вывод

out.txt

```
########################################
2019-12-05 20:29:00
Nothing happened
########################################
2019-12-05 20:30:00
The freight train №2 is currently arriving
5 carriages were added to the train №5 at 2019-12-05 20:30:00
########################################
2019-12-05 20:31:00
Nothing happened
########################################
```

Тестовое покрытие

classes_test.txt

```python
def test_delay_arrival_passenger():
    date = '14/03/2011'
    initialize(date)
    expected = []
    real_data = []

    for _ in repeat(None, 100000):
        time = generate_time()
        arrival_value = date + ' ' + time
        pass_train.set_arrival(arrival_value)

        delta_changes = timedelta(minutes=random.randint(0, 30))
        changes_time = pass_train.arrival - delta_changes
        delta_arrival = random.randint(1, 59)
        pass_train.delay_arrival(int(delta_arrival),
                                 changes_time.strftime('%d/%m/%Y %H:%M'))
        expected.append(PASSENGER_DELAY_ARRIVAL.format(
                                 pass_train.train_id,
                                 delta_arrival))

        for event in schedule[changes_time]:
            real_data.append(event)

    assert set(expected) == set(real_data)
```

Что улучшить?

- Logging и обработка ошибок / исключений
- Базы данных
- Новые события…

Спасибо за внимание