Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ

ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет Безопасности Информационных Технологий

Дисциплина

«Управление мобильными устройствами»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«Обработка и тарификация CDR (Call Detail Record)»

Вариант 2

Выполнила:

Студентка группы N3350

Шкарева Алена Дмитриевна



Проверил:

доцент ФБИТ,

Федоров Иван Романович

Санкт-Петербург

2020

# Цель работы:

Изучение биллинговой системы и тарификации звонков.

# Задачи:

Реализация простейшего правила тарификации для услуг типа “Телефония” по длительности разговора и “СМС” по общему количеству.

# Реализация:

Для реализации программы был выбран язык Python.

Классы ‘Coefficient’ были реализованы, чтобы для каждой подуслуги использования телефонии (исходящие и входящие звонки, СМС) можно было задать собственные правила расчета.

Классы типа ‘Rule’ – правила расчета также имеют возможные типы – по количеству, по периоду, а также базовый для расчета записей, не входящих в дополнительные правила.

Класс ‘Tariff’ производит подготовку записей для расчетов каждой из подуслуг.

Класс ‘BaseBilling’ нужен для хранения информации об абоненте, его номер, записи и используемый тариф.

Класс ‘DataManager’ используется для перевода данных об использовании услугой разных абонентов к тому виду, который используется программой.

Из сторонних библиотек используется библиотека pandas.

Выполнение программы:



# Выводы

Во время выполнения данной лабораторной работы были реализованы простейшие правила тарификации.

# Исходный код

<https://github.com/gammilen/mobile_device_management>

**program.py**

**from** billing **import** BaseBilling

**from** tariffs **import** Tariff

**from** coefficients **import** IncomingCoefficient**,** OutgoingCoefficient**,** SMSCoefficient

**from** rules **import** BasicRule

**from** data\_manager **import** DataManager

**def** calculate\_tel**():**

i **=** IncomingCoefficient**()**

i**.**set\_base\_rule**(**BasicRule**(**1**))**

o **=** OutgoingCoefficient**()**

o**.**set\_base\_rule**(**BasicRule**(**3**))**

s **=** SMSCoefficient**()**

s**.**set\_base\_rule**(**BasicRule**(**1**))**

t **=** Tariff**(**i**,** o**,** s**)**

b **=** BaseBilling**(**"968247916"**)**

b**.**load\_tariff**(**t**)**

b**.**load\_records**(**DataManager**.**get\_data**())**

**return** b**.**count**()**

**def** main**():**

**print(**"Результат: "**,** calculate\_tel**())**

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

main**()**

**data\_manager.py**

**import** pandas **as** pd

**import** os

file\_name **=** os**.**path**.**join**(**os**.**path**.**dirname**(**os**.**path**.**abspath**(**\_\_file\_\_**)),** "data.csv"**)**

**class** **DataManager:**

@classmethod

**def** get\_data**(**cls**,** f\_name**=**file\_name**):**

data **=** pd**.**read\_csv**(**f\_name**)**

data**[**'msisdn\_origin'**]** **=** data**[**'msisdn\_origin'**].**astype**(**str**)**

data**[**'msisdn\_dest'**]** **=** data**[**'msisdn\_dest'**].**astype**(**str**)**

**return** data

**coefficients.py**

**from** abc **import** ABC**,** abstractmethod

**class** **Restriction:**

**def** \_\_init\_\_**(**self**,** **\***args**,** **\*\***kwargs**):**

self**.**count **=** **{**

"free finite"**:** **True,**

"free amount"**:** 1**,**

**}**

self**.**period **=** **{**

"free finite"**:** **False,**

"free amount"**:** **None,**

**}**

self**.**validation **=** **{**

"free-count"**:** self**.**free\_count**,**

"free-period"**:** self**.**free\_period**,**

"non-free-count"**:** self**.**non\_free\_count**,**

"non-free-period"**:** self**.**non\_free\_period**,**

**}**

#вызывается при добавлении non-free-count правила

@staticmethod

**def** non\_free\_count**(**period\_num**,** free\_period\_num**):**

**if** period\_num **!=** free\_period\_num**:**

**raise** ValueError**(**"Нарушен порядок сочетания правил"**)**

#вызывается при добавлении non-free-period правила

@staticmethod

**def** non\_free\_period**(**count\_num**,** free\_count\_num**):**

**if** count\_num **!=** free\_count\_num**:**

**raise** ValueError**(**"Нарушен порядок сочетания правил"**)**

**def** free\_count**(**self**,** free\_count\_num**):**

**if** self**.**count**[**'free finite'**]:**

**if** free\_count\_num **!=** self**.**count**[**'free amount'**]:**

**raise** ValueError**(**

"Нарушен порядок использования"

"правил типа 'количество' с отсутствием платы"**)**

**def** free\_period**(**self**,** free\_period\_num**):**

**if** self**.**period**[**'free finite'**]:**

**if** free\_period\_num **!=** self**.**count**[**'free amount'**]:**

**raise** ValueError**(**

"Нарушен порядок использования"

"правил типа 'количество' с отсутствием платы"**)**

#TODO Проверка на пересечение

# периоды в формате (начало, конец)

**def** period**(**self**,** periods**):**

**pass**

**class** **Coefficient(**ABC**):**

**def** \_\_init\_\_**(**self**,** **\***args**,** **\*\***kwargs**):**

self**.**restriction **=** Restriction**()**

self**.**free\_period\_num **=** 0

self**.**free\_count\_num **=** 0

self**.**period **=** dict**()**

self**.**count **=** dict**()**

**def** set\_base\_rule**(**self**,** rule**):**

self**.**base **=** rule

**class** **CallingCoefficient(**Coefficient**):**

**def** \_\_init\_\_**(**self**,** **\***args**,** **\*\***kwargs**):**

super**(**CallingCoefficient**,** self**).**\_\_init\_\_**(\***args**,** **\*\***kwargs**)**

self**.**free\_period\_num **=** 0

self**.**free\_count\_num **=** 0

self**.**period **=** dict**()**

self**.**count **=** dict**()**

**def** add\_period\_rule**(**self**,** rule**):**

**if** rule**.**k **==** 0**:**

self**.**restriction**.**validation**[**'free-period'**](**self**.**free\_period\_num**+**1**)**

self**.**free\_period\_num **+=** 1

self**.**update\_rules**(**rule**,** "period"**,** **True)**

**else:**

self**.**restriction**.**validation**[**'non-free-period'**](**

len**(**self**.**count**),** self**.**free\_count\_num**)**

self**.**update\_rules**(**rule**,** "period"**,** **False)**

**def** add\_count\_rule**(**self**,** rule**):**

**if** rule**.**k **==** 0**:**

self**.**restriction**.**validation**[**'free-count'**](**self**.**free\_count\_num**+**1**)**

self**.**free\_count\_num **+=** 1

self**.**update\_rules**(**rule**,** "count"**,** **True)**

**else:**

self**.**restriction**.**validation**[**'non-free-count'**](**

len**(**self**.**period**),** self**.**free\_period\_num**)**

self**.**update\_rules**(**rule**,** "count"**,** **False)**

**def** has\_free**(**self**):**

**return** self**.**free\_count\_num **>** 0 **or** self**.**free\_period\_num **>** 0

**def** update\_rules**(**self**,** \_rule**,** r\_type**,** free**):**

**if** r\_type **==** "period"**:**

**if** free **is** **True:**

self**.**period**[**"free"**].**append**(**\_rule**)**

**else:**

self**.**period**[**"non-free"**].**append**(**\_rule**)**

**elif** r\_type **==** "count"**:**

**if** free **is** **True:**

self**.**count**[**"free"**].**append**(**\_rule**)**

**else:**

self**.**count**[**"non-free"**].**append**(**\_rule**)**

#TODO Учет прочих правил при расчете

**def** calculate**(**self**,** recordset**):**

# return self.base \* (sum of out calling)

**return** self**.**base**.**k **\*** recordset**[**"call\_duration"**].**sum**()**

**class** **IncomingCoefficient(**CallingCoefficient**):**

**pass**

**class** **OutgoingCoefficient(**CallingCoefficient**):**

**pass**

**class** **SMSCoefficient(**Coefficient**):**

**def** \_\_init\_\_**(**self**,** **\***args**,** **\*\***kwargs**):**

super**(**SMSCoefficient**,** self**).**\_\_init\_\_**(\***args**,** **\*\***kwargs**)**

self**.**free\_count\_num **=** 0

self**.**count **=** dict**()**

**def** add\_count\_rule**(**self**,** rule**):**

**if** rule**.**k **==** 0**:**

self**.**restriction**.**validation**[**'free-count'**](**self**.**free\_count\_num**+**1**)**

self**.**free\_count\_num **+=** 1

self**.**update\_rules**(**rule**,** "count"**,** **True)**

**else:**

self**.**restriction**.**validation**[**'non-free-count'**](**

len**(**self**.**period**),** self**.**free\_period\_num**)**

self**.**update\_rules**(**rule**,** "count"**,** **False)**

**def** has\_free**(**self**):**

**return** self**.**free\_count\_num **>** 0

**def** update\_rules**(**self**,** \_rule**,** r\_type**,** free**):**

**if** r\_type **==** "count"**:**

**if** free **is** **True:**

self**.**count**[**"free"**].**append**(**\_rule**)**

**else:**

self**.**count**[**"non-free"**].**append**(**\_rule**)**

#TODO Учет прочих правил при расчете

**def** calculate**(**self**,** recordset**):**

# return self.base \* (sum of sms)

**return** self**.**base**.**k **\*** recordset**[**"sms\_number"**].**sum**()**

**billing.py**

**class** **BaseBilling:**

**def** \_\_init\_\_**(**self**,** subscriber**,** **\***args**,** **\*\***kwargs**):**

self**.**records **=** **None**

self**.**tariff **=** **None**

self**.**subscriber **=** subscriber

**def** load\_records**(**self**,** recordset**):**

self**.**records **=** recordset

**def** load\_tariff**(**self**,** tariff**):**

tariff**.**load\_subscriber**(**self**.**subscriber**)**

self**.**tariff **=** tariff

**def** count**(**self**):**

**if** **not** self**.**tariff**:**

**raise** AttributeError**(**"Отсутствует тариф для расчета"**)**

**return** self**.**tariff**.**calculate**(**self**.**records**)**

**tariffs.py**

**class** **Tariff:**

**def** \_\_init\_\_**(**self**,** incoming**,** outgoing**,** sms**):**

self**.**coeffs **=** **{**

'incoming'**:** incoming**,**

'outgoing'**:** outgoing**,**

'sms'**:** sms

**}**

**def** load\_subscriber**(**self**,** sub**):**

self**.**subscriber **=** sub

**def** \_filter\_incoming**(**self**,** recordset**):**

**return** recordset**[**recordset**[**"msisdn\_dest"**]** **==** self**.**subscriber**][[**"timestamp"**,**"call\_duration"**]]**

**def** \_filter\_outgoing**(**self**,**recordset**):**

**return** recordset**[**recordset**[**"msisdn\_origin"**]** **==** self**.**subscriber**][[**"timestamp"**,**"call\_duration"**]]**

**def** \_filter\_sms**(**self**,** recordset**):**

**return** recordset**[**recordset**[**"msisdn\_origin"**]** **==** self**.**subscriber**][[**"timestamp"**,**"sms\_number"**]]**

**def** \_calculate\_incoming**(**self**,** recordset**):**

**return** self**.**coeffs**[**"incoming"**].**calculate**(**recordset**)**

**def** \_calculate\_outgoing**(**self**,** recordset**):**

**return** self**.**coeffs**[**"outgoing"**].**calculate**(**recordset**)**

**def** \_calculate\_sms**(**self**,** recordset**):**

**return** self**.**coeffs**[**"sms"**].**calculate**(**recordset**)**

**def** calculate**(**self**,** recordset**):**

result **=** 0

result **+=** self**.**\_calculate\_incoming**(**self**.**\_filter\_incoming**(**recordset**))**

result **+=** self**.**\_calculate\_outgoing**(**self**.**\_filter\_outgoing**(**recordset**))**

result **+=** self**.**\_calculate\_sms**(**self**.**\_filter\_sms**(**recordset**))**

**return** result

**rules.py**

**from** abc **import** ABC

**class** **Rule(**ABC**):**

**def** \_\_init\_\_**(**self**,** k**,** **\***args**,** **\*\***kwargs**):**

self**.**k **=** k

**class** **BasicRule(**Rule**):**

#Просто стабильный множитель

**pass**

**class** **PeriodRule(**Rule**):**

#Ограничения по времени

#Если не указано то 'до' с наименьшего времени, если 'после', то до наибольшего времени

**pass**

**class** **CountRule(**Rule**):**

#Множитель зависит от количества

#Если не указано 'от', то с 0 , если не 'до', то бесконечности

**pass**