CS542 Project Three

Xiaoyan Sun & Shi Wang

1. Design and Assumptions

In this project, we simulated a query execution engine using the iterator framework and observer pattern. The query executed are expressed as the following in SQL:

SELECT City.Name
From City, County
Where City.CountryCode=Country.Code
AND City.Population > Country.Population

To execute this query, three operators are required, which are join, selection and projection. Here are some of the key points and assumptions:

- Relations are read in from .csv files, and stored in memory as ArrayList of tuples
- Tuples are stored in String[]
- The join function will produce, in an iterating manner, joined tuples of the two relations City and Country on their common attribute countrycode/code. For each call of the GetNext() in join operation, a new tuple of joined attributes is returned.
- The selection operation will select tuples that meet the requirement that city population is more than 40% of country population.
- Observer pattern is used between operator join and select. The select operator is notified when a new tuple is outputted by the join operator.
- Operations are executed in sequential order and the output is to print on screen, so no writing back to disk is necessary.

2. Algorithm

2.1 the relation class

The relation class contains three methods: Open(), GetNext() and Close()
Open(): read in data from .csv file and store them in the ArrayList of tuples
GetNext(): return the first tuple in the ArrayList, remove the previous processed tuple from the ArrayList.

GetNext():

```
public tuple GetNext(){

                    //Get the next tuple in the relation
                                        if(!tuples.isEmpty()){
                                                  tuple t= tuples.get(0);
                                                  tuples.remove(0);
                                                  return t;
```

```
        }
    else{
        return null;
        }
    }
```

2.4 Join

- We join two relations in the new tuples for the selection method.
- We use iterator methods for tuple-based nested-loop join of two relations. We repeatedly read two relations city and country.

the join algorithm:

```
Open(){
    R.Open();
    S.Open();
    s=S.GetNext();
}
GetNext(){
    REPEAT{
        r=R.GetNext();
        IF(r=NotFound){
            R.Close();
            s=S.GetNext();
            IF(s=NotFound){ Return NotFound;}
            R.Open();
            r.GetNext();
        }
    }UNTIL(r and s join);
RETURN the join tuple of r and s;
}
Close(){
    R.Close();
    S.Close();
}
```

2.5 Select & Project

- We select the new tuples after join to find the required tuples.
- Then we project the required attribute.

abstract of code:

```
//do the selection with an observer
public void update(Observable o, Object arg)
t = (tuple) arg;
//Choose the required tuples by the condition
if(Integer.parseInt(t.getOthers()[4]) > Integer.parseInt(t.getOthers()[10])*0.4)
```

3. test and result

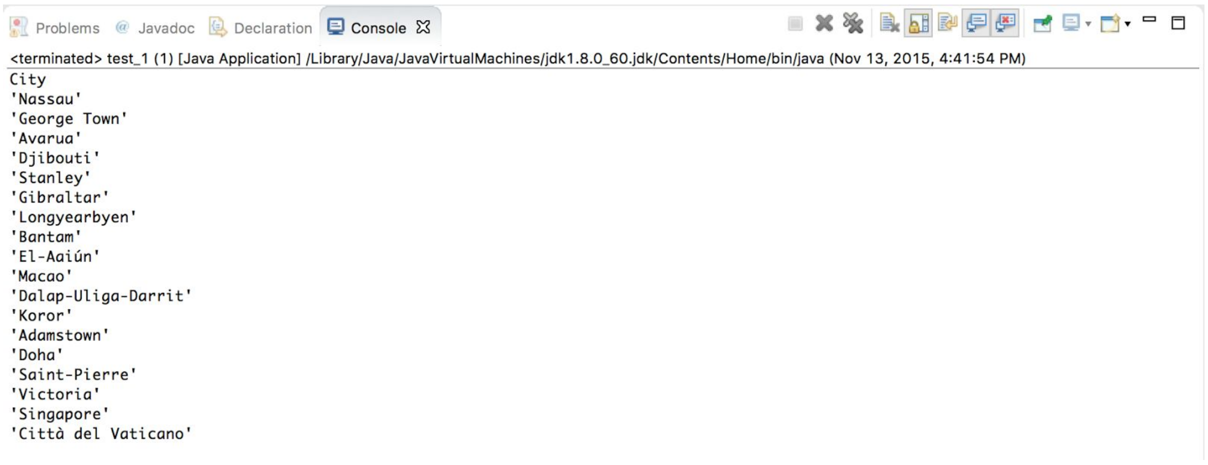
3.1 Use the class to find all cities whose population is more than 40% of the population of their entire country.

<u>(1)method and process</u>

- We join the city and country tables with the same country code into new tuples.
- We select the tuples where the city's population is more than 40% of the country's population.
- We project the attribute city name.

<u>(2)result</u>

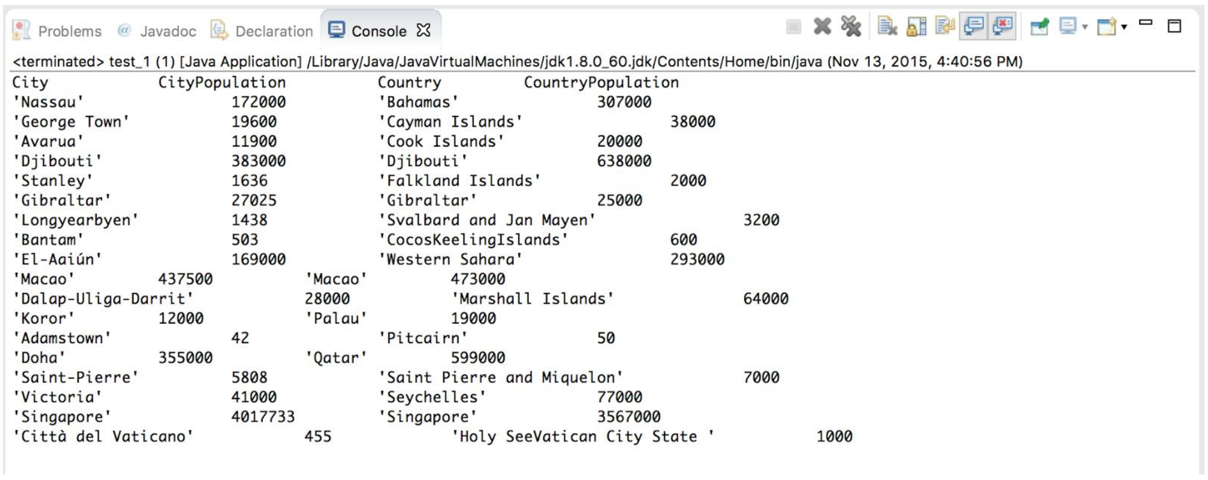
- We get the cities whose population is more than 40% of the country's population.



```
<terminated> test_1 (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (Nov 13, 2015, 4:41:54 PM)
City
'Nassau'
'George Town'
'Avarua'
'Djibouti'
'Stanley'
'Gibraltar'
'Longyearbyen'
'Bantam'
'El-Aaiún'
'Macao'
'Dalap-Uliga-Darrit'
'Koror'
'Adamstown'
'Doha'
'Saint-Pierre'
'Victoria'
'Singapore'
'Città del Vaticano'
```

<u>(3)correctness</u>

- The number of the city's population is more than 40% of the country's population.



```
<terminated> test_1 (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (Nov 13, 2015, 4:40:56 PM)
City            CityPopulation          Country         CountryPopulation
'Nassau'                172000          'Bahamas'               307000
'George Town'           19600           'Cayman Islands'                38000
'Avarua'                11900           'Cook Islands'          20000
'Djibouti'              383000          'Djibouti'              638000
'Stanley'               1636            'Falkland Islands'              2000
'Gibraltar'             27025           'Gibraltar'             25000
'Longyearbyen'          1438            'Svalbard and Jan Mayen'                3200
'Bantam'                503             'CocosKeelingIslands'           600
'El-Aaiún'              169000          'Western Sahara'                293000
'Macao'         437500          'Macao'         473000
'Dalap-Uliga-Darrit'            28000           'Marshall Islands'              64000
'Koror'         12000           'Palau'         19000
'Adamstown'             42              'Pitcairn'              50
'Doha'          355000          'Qatar'         599000
'Saint-Pierre'          5808            'Saint Pierre and Miquelon'             7000
'Victoria'              41000           'Seychelles'            77000
'Singapore'             4017733         'Singapore'             3567000
'Città del Vaticano'            455             'Holy SeeVatican City State '           1000
```