Project 3 Document

Author: Yan Wang & Zishan Qin

Project Requirement

1. Use the class to create and populate city and country tables
2. Use the class to find all cities whose population is more than 40% of the population of their entire country.

Assumption

1. The memory can contain 10 blocks of each relation.
2. One block contains 10 records of a relation. We assume this way just to mimic the case that the memory can not hold the entire table. However, a more reasonable assumption would be “One block contains 1k bytes”, which is not addressed in this project.
3. We treat the csv files as relation table directly, rather than create and populate city and country tables. However, in order to show that our program has the ability to create and populate a new relation, we generate a “New.csv” file to test the functionality.

Design

1. There are 3 classes: Main.java, Relation.java and JoinRelation.java.
2. Class Main is the runnable file, and used to test the other two classes. In this class, we tested 2 functionality：
    a. Join of relation “city” and “county”.
    b. Test the functionality to create and populate a relation.
3. Class Relation is used to create a relation object. Methods in this class is as follows:
    a. public Relation(String relationName) Constructor of the relation. If the relation file exists, open it; otherwise, creates a new file for this relation.
    b. public void put(String record) Put record in the relation
    c. public int open(int numBlocks) Read in several (numBlocks) blocks in memory
    d. public String [] getNext() Get next tuple from the blocks that have been read in
    e. public void resetNext() Point to the beginning of the blocks that have been read in.
    f. public void resetBlock() Point to the beginning of the relation
    g. public void close() Close the relation
4. Class JoinRelation is used to implement join of two relations. Methods in this class include:
    a. public JoinRelation(Relation left, Relation right, int popLeft, int popRight,int countryLeft, int countryRight, int indexResult, int numBlocks) Accept two relations and relevant parameters for join.

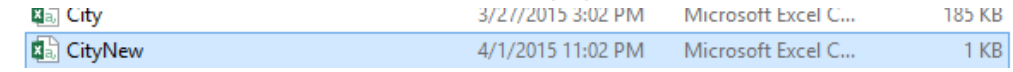
b. public ArrayList<String> simpleJoin() Go through two relations to implement join.

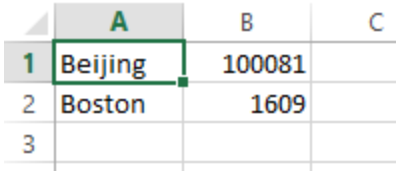
Test Results

1. According to Requirement 1, we tested the functionality to create and populate a relation.
   TEST:

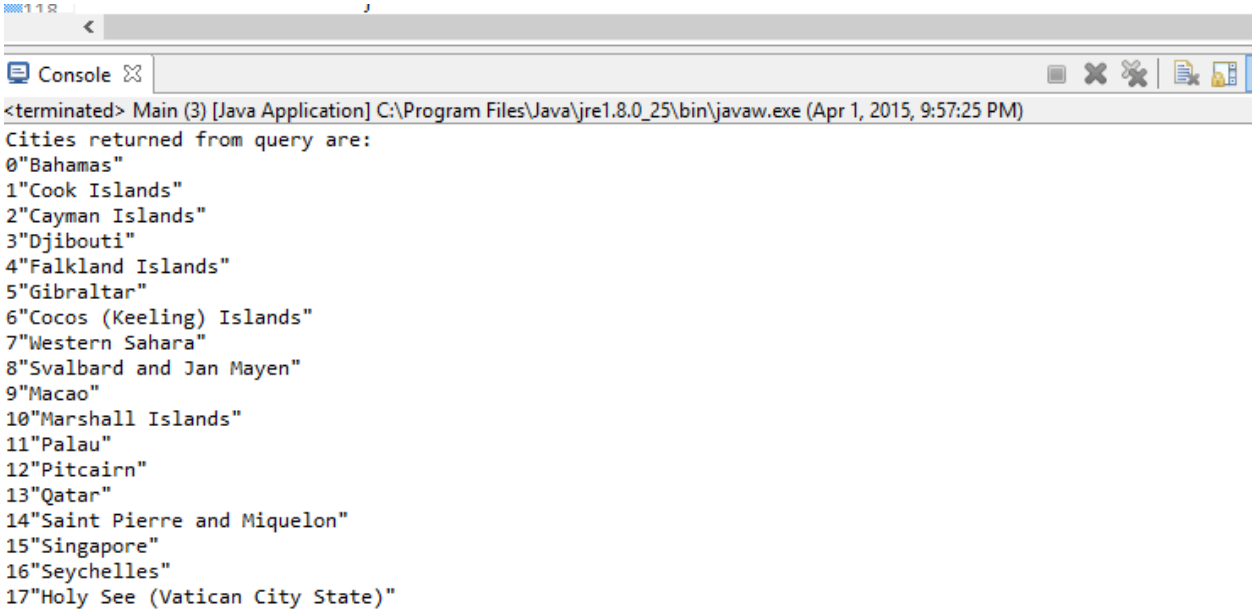
```
//Relation try create and write to a relation
Relation newRelation = new Relation("CityNew.csv");
String newRecord1 = "\"Beijing\",\"100081\"";
String newRecord2 = "\"Boston\",\"01609\"";
newRelation.put(newRecord1);
newRelation.put(newRecord2);
```

   RESULTS: A new relation is created and populated.





2. According to Requirement 2, we tested the join of relation city and country. 18 records satisfies the requirement. The screenshot the output results are shown below.


Console
<terminated> Main (3) [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (Apr 1, 2015, 9:57:25 PM)
Cities returned from query are:
0"Bahamas"
1"Cook Islands"
2"Cayman Islands"
3"Djibouti"
4"Falkland Islands"
5"Gibraltar"
6"Cocos (Keeling) Islands"
7"Western Sahara"
8"Svalbard and Jan Mayen"
9"Macao"
10"Marshall Islands"
11"Palau"
12"Pitcairn"
13"Qatar"
14"Saint Pierre and Miquelon"
15"Singapore"
16"Seychelles"
17"Holy See (Vatican City State)"