1. Continue lab 10 to do lab 11
2. Create a class named FileDAO. This class contains some static methods

/\* this method will each line data from a text file, converts it to a student object and add the object to ArrayList

input: a filename

output: a HashMap is a map , mapping the campus name to list of students of that campus. \*/

**public static HashMap<String, List<Student>> readFile(String filename){**

**//TODO:**

**}**

/\*This method to write a HashMap to text file

Input: a filename, a HashMap. Mapping campus name to list of students of that campus

Output: true/false

\*/

**public boolean writeFile(String filename, HashMap<String, ArrayList<Student>> list)**

**{**

**//TODO:**

**}**

1. Add some methods to StudentManager

class **StudentManager**{

HashMap<String, List<Student>> h;

….

**//this method to call readFile()  
 public void loadStudents(){   
 String filename=”data.txt”;  
 h= FileDAO.readFile(filename);  
 }**

**//this method to call writeFile**

**public void backup(){**

**String filename=”backup.txt”;  
 if ( FileDao.writeFile(filename))**

**System.out.println(“saved!!!”);  
 else System.out.println(“save fail”);**

**}**

3. Create a class named “Tester”, it contains the method main to call all methods above

Menu:

1. Load Students
2. Backup
3. Exit

Assignment: Website Visits

In this assignment you will build on the LogEntry and LogAnalyzer classes that you worked on in the last lesson. You will write several methods to solve problems about web logs.

There are four small files you can use to test the methods: short¬test\_log, weblog¬short\_log, weblog2-short\_log, and weblog3¬short\_log. You should write code to test the methods in a Tester class. In tester class, you create a LogAnalyzer object.

Specifically, you should do the following:

● In the LogAnalyzer class, write the method countVisitsPerIP, which has one input parameters that is a filename. This method returns a HashMap that maps an IP address to the number of times that IP address appears in records,meaning the number of times this IP address visited the website. Recall that records stores LogEntrys from a file of web logs

● In the LogAnalyzer class, write the method mostNumberVisitsByIP, which has one parameter, a HashMap that maps an IP address to the number of times that IP address appears in the web log file. This method returns the maximum number of visits to this website by a single IP address. For example, the call mostNumberVisitsByIPon a HashMap formed using the file weblog3¬short\_log returns 3.

● In the LogAnalyzer class, write the method iPsMostVisits, which has one parameter, a HashMap that maps an IP address to the number of times that IP address appears in the web log file. This method returns an ArrayList of Strings of IP addresses that all have the maximum number of visits to this website. For example, the Java Programming: Arrays, Lists, and Structured Data call iPsMostVisitson a HashMap formed using the file weblog3¬short\_log returns the ArrayList with these two IP addresses, 61.15.121.171 and 84.133.195.161. Both of them visited the site three times, which is the maximum number of times any IP address visited the site.

● In the LogAnalyzer class, write the method iPsForDays, which has no parameters. This method returns a HashMap that uses records and maps days from web logs to an ArrayList of IP addresses that occurred on that day. A day is in the format “MMM DD” where MMM is the first three characters of the month name with the first letter capital and the others in lowercase, and DD is the day in two digits (examples are “Dec 05” and “Apr 22”). For example, for the file weblog3¬short\_log, after building this HashMap, if you print it out, you will see that Sep 14 maps to one IP address, Sep 21 maps to four IP addresses, and Sep 30 maps to five IP addresses.

● In the LogAnalyzer class, write the method dayWithMostIPVisits, which has one parameter that is a HashMap that uses records and maps days from web logs to an ArrayList of IP addresses that occurred on that day. This method returns the day that has the most IP address visits. If there is a tie, then return any such day. For example, if you use the file weblog3¬short\_log,then this method should return the day most visited as Sep 30.

● In the LogAnalyzer class, write the method iPsWithMostVisitsOnDay, which has two parameters—the first one is a HashMap that uses records and maps days from web logs to an ArrayList of IP addresses that occurred on that day, and the second parameter is a String representing a day in the format “MMM DD” described above. This method returns an ArrayList of IP addresses that had the most accesses on the given day. For example, if you use the file weblog3¬short\_log, and the parameter for the day is “Sep 30”, then there are two IP addresses in the ArrayList returned: 61.15.121.171 and 177.4.40.87. Hint: This method should call another method you have written.

Hint:

public class LogEntry

{

private String ipAddress;

private String accessTime;

private String request;

private int statusCode;

private int bytesReturned;

TODO: add constructors/getters/setters in this class

}

public class LogAnalyzer

{

//dem so lan truy cap web cua tung IP

public HashMap<String, Integer> countVisitsPerIP(String filename){

// TODO: add some methods

}

Example: using file: short-test\_log

|  |  |
| --- | --- |
| Key | Value |
| 110.76.104.12 | 1 |
| 152.3.135.44 | 3 |
| 157.55.39.203 | 1 |
| 152.3.135.63 | 2 |

//tim so lan truy cap web nhieu nhat

public dayWithMostIPVisits(HashMap<String,Integer> list){

//TODO

}

Example: using short-test\_log will return 3.

// maps days from web logs to an ArrayList of IP addresses that occurred on that day.

Public HashMap<String, ArrayList<String>> iPsMostVisits(){  
 //TODO

}

Example: file weblog2-short\_log

|  |  |
| --- | --- |
| Key | Value |
| 21/Sep/2015 | 84.189.158.117, 61.15.121.171, 84.133.195.161, 177.4.40.87, 84.133.195.161 |
| 30/Sep/2015 | 84.189.158.117, 61.15.121.171, 61.15.121.171, 177.4.40.87, 177.4.40.87 |
| 14/Sep/2015 | 84.133.195.161 |

….

**Assignment: AutoComplete**

This assignment will read a text file (words.small) to create a vocabulary. A HashMap is used to store all words from the file.

Example:

|  |  |
| --- | --- |
| Key | Value |
| “a” | A, abilities, ability, ability's, able, about, above, absence, absolute, absolutely, abuse, academic, accept, ….. |
| “b” | “book”,”baby”,… |
| … | …. |

Requirements:

public class Vocabulary{

HashMap<String, ArrayList<String> voc;

//constructor

….

//load all words from the text file to voc

//use the method **addWords**(String key, ArrayList<String> list) to do

public void loadWords(String filename)

{

//TODO;

}

/\*User inputs a String (prefix) and a number (limit words for support) and this methods will display some words( in limit) that contains prefix

input prefix =”a”, and limit=6

output: a, abilities, ability, ability's, able, about

\*/

public void autoComplete()

//TODO

}

/\*this method to check a word which is real word.   
 Means: a real word is word which exists in HashMap\*/  
public boolean isRealWord(String word){

//TODO

}

/\*This method will get a word which has the max length\*/

public String WordMaxLength(){

//TODO

}

A class Tester to test all methods