Report

Assignment 8 : Hash Map

จัดทำโดย

นายสรธร แก้วโชติช่วงกูล 63070501067 CPE REGULAR

รายงานนี้เป็นส่วนหนึ่งของวิชา CPE111

Programming With Data Structures

King Mongkut's University of Technology Thonburi

- **สิ่งที่ทำใน Assignment**

1. สร้าง class ในการทำงานเกี่ยวกับ Dictionary(HashMap) ประกอบด้วย methods ได้แก่
    - Method ReadFile เอาไว้อ่านไฟล์แล้วเพิ่มข้อมูลเข้าไปใน HashMap
    - Method AddHashMap เอาไว้เช็คข้อมูลก่อนเพิ่มเข้าไปใน Hash
    - Method Show_stats โชว์สถิติของข้อมูลต่างๆ
    - Method Find_words เอาไว้หา word ที่ผู้ใช้ต้องการหา
    - Method CompareMean เอาไว้เช็ค mean ว่ามีใน Hash แล้วหรือยัง
2. สร้าง class ในการทำงานเกี่ยวกับ Structure(โครงสร้างข้อมูล) โดยเฉพาะและสร้าง method ได้แก่
    - constructor(มีข้อมูล) การตรวจสอบแล้วเพิ่ม mean & type เข้าไปใน ArrayList ของ Structure

- Source code

- Class DictHash(main)

➢ method Read_File

```java
import java.io.BufferedReader;

public class DictHash {
    static HashMap <String, Dnode> dict = new HashMap<String, Dnode> (); // reserve hashmap
    public static int Read_File() {
        int count1=0;
        String str;
        try {
            FileInputStream fcsv = new FileInputStream("C:\\Users\\LENOVO\\Desktop"
                    + "\\CPE-YEAR#1\\CPE SUPJECT\\CPE111 DATA STRUCTURE\\Assignment 6\\UTF8_Lexitron.csv"); // สร้างไฟล์มาเก็บตัวแปรที่ระบุ Endcoding
            InputStreamReader utf = new InputStreamReader(fcsv,"UTF-8"); // ระบุ Endcoding
            BufferedReader buff = new BufferedReader(utf); // ใช้ class bufferreader อ่านข้อมูล BOM ทิ้ง
            buff.read(); // อ่าน BOM ทิ้ง (BOM มีขนาดเท่ากับ char เลยใช้ .read() เหมือนอ่าน char )
            while((str = buff.readLine()) != null ) { // วนรอบอ่านไฟล์
                AddHashMap(str); // check node have been ?
                count1++; // นับจำนวน
            }
            fcsv.close(); // ปิดไฟล์เมื่ออ่านเสร็จ
        }
        catch(Exception e){
            System.out.println("\t\t\tOperation Error!."); // if error
        }
        return count1; // total read
    }
```

➢ method CompareMean

```java
    public static boolean Compare_Mean(Dnode z,String mean) { // use to compare different mean
        int i;
        boolean check = false;
        for(i=0;i<z.mean.size();i++) { // loop
            if(z.mean.get(i).equalsIgnoreCase(mean)) // check every mean in that key
                check = true; // if mean have already
        }
        return check; // then return value of check
    }
```

➢ Method AddHashMap

```java
    public static void AddHashMap(String str) {
        String key;
        str = str.trim().toLowerCase().replaceAll("\\s+"," "); // ***** important don't forget to convert String to lower or upper
        String [] buff = str.split(","); // split มันจาก ,
        key = buff[0]; // first of buff is always word
        Dnode x = new Dnode(buff); // reserve new Node to add mean
        if(dict.containsKey(key)) { // check key ever in Hash ?
            Dnode z = dict.get(key); // create a new variable to compare and add mean
            if(!Compare_Mean(z,x.mean.get(0))){ // if mean isn't ever in hash
                z.mean.add(x.mean.get(0)); // add mean into ArrayList of Node
            }
        }
        else { // if key isn't ever in hash
            dict.put(key,x); // put new Node in hash
        }
    }
```

➢ method show_stats

```java
    public static void Show_stats(int count) {
        int max = 0,sum=0,i;
        String mostword = null;
        System.out.println("\t\t\tTotal Read = "+count+" records.");
        System.out.printf("\t\t\tNumber of KeyWord : %d records.\n",dict.size());
        for(String itr : dict.keySet()) { // loop to find sum of mean & max keywords
            Dnode value = dict.get(itr);
            sum = sum + value.mean.size();
            if(max < value.mean.size()) {
                max = value.mean.size();
                mostword = itr;
            }
        }
        System.out.printf("\t\t\tNumber of Meaning : %d records.\n",sum);
        System.out.printf("\t\t\tMost Keywords is %s Found : %d records.\n",mostword,max);
        Dnode z = dict.get(mostword); // create new variable to show meaning in Node
        for(i=1;i<=max;i++) {
            System.out.printf("\t\t\t%d ) %s\n",i,z.mean.get(i-1));
        }
    }
```

➢ Method Find_words

```java
    public static void Find_words(String str) { // Find Keywords in Hash
        int i;
        if(dict.containsKey(str)) { // check keyword is in hash?
            Dnode z = dict.get(str);
            for(i=1;i<=z.mean.size();i++) { // if keyword have been in hash
                System.out.printf("\t\t\t%d ) %s\n",i,z.mean.get(i-1)); // show all mean of that word
            }
        }
        else System.out.printf("\t\t\tDon't have %s in HashMap!.\n",str); // if keyword haven't been in hash
    }
```

➢ Main

```java
    public static void main(String[] args) {
        int count;
        String str = "";
        count = Read_File(); // ReadFile & count total Read
        Show_stats(count); // show every stats in testcase
        Scanner in = new Scanner(System.in);
        while(!str.equalsIgnoreCase("end")){ // loop to Read Keyword if doesn't match "end"
            System.out.print("Enter your words >  ");
            str = in.nextLine(); // Read keywords
            str = str.trim().toLowerCase().replaceAll("\\s+"," "); //************ important don't forget to convert String to lower or upper
            Find_words(str); // then put it in FindWord method
        }
        System.out.printf("\t\tEndProgram.\n");
        System.out.printf("\t\tThis program is writen by Sorathorn Kaewchotchuangkul 63070501067 CPE/1");
    }

}
```

- Class ที่ทำงานเกี่ยวกับข้อมูล (Dnode)

```java
import java.util.ArrayList;

public class Dnode {
    ArrayList<String> mean;
    public Dnode() {
        mean = new ArrayList<String>();
        mean.add("");
    }
    public Dnode(String [] buff) {
        String meaning;
        if(buff.length == 1) // only words
            meaning = "";
        else if(buff.length == 2) { // have meaning
            meaning = buff[1];
        }
        else // have meaning & type
            meaning = buff[1] + "("+ buff[2] +")";
        mean = new ArrayList<String>();
        mean.add(meaning);
    }
}

```

Note! อธิบายโค้ด comment อยู่ในตัวโค้ด

- Test case & อธิบาย

```
Total Read = 74233 records.
Number of KeyWord : 45921 records.
Number of Meaning : 73981 records.
Most Keywords is get off Found : 35 records
1 ) เริ่ม(phrv)
2 ) เริ่ม (บางสิ่ง) ได้ดีหรือไม่ดี(phrv)
3 ) เริ่มเป็นจริง(phrv)
4 ) เริ่มต้นดี(phrv)
5 ) เริ่มทำงาน(phrv)
6 ) เรียนรู้(phrv)
7 ) เลิกเย่อหยิ่ง(phrv)
8 ) เลิกงาน(phrv)
9 ) เลิกทำให้ฉันรำคาญ(phrv)
10 ) เอาออกไป(phrv)
11 ) แต่งงาน(phrv)
12 ) โน้มลงมา(phrv)
13 ) ไปให้พ้น(phrv)
14 ) ไม่เชื่อหรอก(phrv)
15 ) ไม่โดนลงโทษ(phrv)
16 ) ช่วยเหลือให้ออกมาจาก (เรือที่กำลังจม)(phrv)
17 ) ตื่นเต้น(phrv)
18 ) ทำความสะอาด(phrv)
19 ) ทำตลก(phrv)
20 ) บอกให้เลิกทำหรือไม่ทำ(phrv)
21 ) ปัดความรับผิดชอบ(phrv)
22 ) พัก(phrv)
23 ) มีวันหยุด(phrv)
24 ) ยกลงมา(phrv)
25 ) ยอมรับ (การควบคุม)(phrv)
26 ) รอดพ้น(phrv)
27 ) รอดพ้นอันตราย(phrv)
28 ) ลงมาจาก(phrv)
29 ) ลืม(phrv)
30 ) ส่ง(phrv)
31 ) หนีรอดจาก(phrv)
32 ) หลับ(phrv)
33 ) ออกเดินทาง(phrv)
34 ) ออกจาก (รถ)(phrv)
35 ) ออกจากรถ(phrv)
```

เมื่อเริ่มโปรแกรมจะเริ่มอ่านไฟล์และนับจำนวนที่อ่านได้ จำนวน **Word & Mean** ใน **HashMap** และโชว์ **word** ที่มี **meaning** เยอะที่สุดใน และแสดง **meaning** ออกมาให้เห็นด้วย

```
Enter your words >  a
                    1 ) อักษรตัวแรกในภาษาอังกฤษ(n)
Enter your words >  zymurgy
                    1 ) การหมักสุรา(n)
```

```
Enter your words >  Gamine
                    1 ) (เด็กหญิง) ซึ่งเล่นซุกซนแบบเด็กชาย(adj)
                    2 ) เด็กหญิงที่ชอบเล่นซุกซนแบบเด็กชาย(n)
Enter your words >  CROON
                    1 ) การฮัมเพลง(n)
                    2 ) ฮัมเพลง(vi)
                    3 ) ฮัมเพลง(vt)
Enter your words >  favorite
                    1 ) ซึ่งเป็นที่โปรดปราน(adj)
                    2 ) คนโปรด(n)
                    3 ) ความนิยมชมชอบ(n)
                    4 ) ตัวเก็ง(n)
```

เมื่อแสดงสถิติต่างๆไปตอนเริ่มแล้วก็จะให้ผู้ใช้ใส่ **word** ที่ต้องการค้นหาใน **HashMap** หลังจากนั้นโปรแกรมจะนำไปค้นหาและแสดงความหมายของ **word** ตัวนั้นออกมาด้วยเหมือนกับ **testcase** นี้

```
Enter your words >       acid    rain
                      1 ) ฝนกรด(n)
```

```
Enter your words >      get    off
                      1 ) เริ่ม(phrv)
                      2 ) เริ่ม (บางสิ่ง) ได้ดีหรือไม่ดี(phrv)
                      3 ) เริ่มเป็นจริง(phrv)
                      4 ) เริ่มต้นดี(phrv)
                      5 ) เริ่มทำงาน(phrv)
                      6 ) เรียนรู้(phrv)
                      7 ) เลิกเย่อหยิ่ง(phrv)
                      8 ) เลิกงาน(phrv)
                      9 ) เลิกทำให้ฉันรำคาญ(phrv)
                      10 ) เอาออกไป(phrv)
                      11 ) แต่งงาน(phrv)
                      12 ) โน้มลงมา(phrv)
                      13 ) ไปให้พ้น(phrv)
                      14 ) ไม่เชื่อหรอก(phrv)
                      15 ) ไม่โดนลงโทษ(phrv)
                      16 ) ช่วยเหลือให้ออกมาจาก (เรือที่กำลังจม)(phrv)
                      17 ) ตื่นเต้น(phrv)
                      18 ) ทำความสะอาด(phrv)
                      19 ) ทำตลก(phrv)
                      20 ) บอกให้เลิกทำหรือไม่ทำ(phrv)
                      21 ) ปัดความรับผิดชอบ(phrv)
                      22 ) พัก(phrv)
                      23 ) มีวันหยุด(phrv)
                      24 ) ยกลงมา(phrv)
                      25 ) ยอมรับ (การควบคุม)(phrv)
                      26 ) รอดพ้น(phrv)
                      27 ) รอดพ้นอันตราย(phrv)
                      28 ) ลงมาจาก(phrv)
                      29 ) ลืม(phrv)
                      30 ) ส่ง(phrv)
                      31 ) หนีรอดจาก(phrv)
                      32 ) หลับ(phrv)
                      33 ) ออกเดินทาง(phrv)
                      34 ) ออกจาก (รถ)(phrv)
                      35 ) ออกจากรถ(phrv)
```

Testcase นี้ก็ไม่มีอะไรต่างจากเดิมมากพียงแต่มีการเพิ่มเว้นวรรคเข้ามาตอนใส่ **word** แต่โปรแกรมก็สามารถตัดเว้นวรรคที่ไม่เกี่ยวข้องออกได้จากการ **trim()** และ **replaceAll()** และยังแปลง **word** ให้เป็นตัวเล็กทุกตัวให้ตรงกับใน **Hashmap** เพื่อจะได้ค้นหาได้ตรงไม่ว่าจะใส่พิมพ์เล็กหรือพิมพ์ใหญ่

Testcase นี้ก็ไม่สามารถค้นหาเจอใน **HashMap** เนื่องจาก **containsKey** ไม่พบตัวที่ต้องการค้นหา ทำให้โปรแกรมแสดงข้อความว่า ไม่เจอ **word** นั้นๆ

```
Enter your words >  cpe
                        Don't have cpe in HashMap!.
```

```
Enter your words >  end
                      1 ) เป้าหมาย(n)
                      2 ) ขอบเขต(n)
                      3 ) ความตาย(n)
                      4 ) ตอนจบ(n)
                      5 ) ส่วนที่เหลือ(n)
                      6 ) ส่วนปลายของวัตถุ(n)
                      7 ) ทำให้สิ้นสุด(vi)
                      8 ) มีผลสรุป(vi)
                      9 ) ทำให้สิ้นสุด(vt)
              EndProgram.
              This program is writen by Sorathorn Kaewchotchuangkul 63070501067 CPE/1
```

Testcase นี้ก็ค้นหาตามปกติแล้วเจอจึงแสดง **meaning** ออกมาและหลังจากนั้นก็จบโปรแกรม เนื่องจาก **Loop** นั้นหยุดทำงานเพราะเงื่อนไขของ **String** ที่รับมานั้นเป็นคำว่า **“end”**

- **สรุปความเข้าใจของตนเอง**

ใน Assignment นี้ต้องสร้าง HashMap เพื่อเก็บข้อมูลของ Dictionary โดยการแบ่งข้อมูลเป็น 2 ส่วนให้ตรงกับโครงสร้างของ HashMap คือ key เป็น Keywords(String) และ mean (ArrayList<String>) โดยที่ mean จะเก็บเก็บ mean & type ของคำนั้นๆ นอกจากนี้ยังต้องคิดในส่วนของการเพิ่มข้อมูลเข้าไปใน HashMap ว่าถ้าเกิดกรณีมี key นั้นอยู่แล้วก็ต้องไม่เพิ่มเข้าไปทับของเดิมแต่เพิ่มเพียงแค่ความหมายเข้าไปแทนแต่ความหมายที่จะเพิ่มเข้าไปก็ต้องไม่ซ้ำกับที่มีอยู่แล้วด้วย ในส่วนของการวนรอบหาคำศัพท์ใน HashMap ไปเรื่อยๆก็ไม่ซับซ้อนเพราะสามารถใช้ ContainsKey ในการค้นหาได้เลยและโปรแกรมจะทำการค้นหาและบอกว่าเจอ/ไม่เจอเราก็เพิ่มกรณีเข้าไปว่าถ้าเจอก็โชว์ mean&type แต่ถ้าไม่เจอก็บอกไม่เจอ แต่ข้อควรระวังในโครงสร้างแบบ HashMap ต้องระวังถึง Capitalization ของตัวอักษรเราต้องแปลงตัวอักษรให้เป็นตัวเล็กหมดหรือตัวใหญ่หมดก่อนที่จะเพิ่มเข้าไปใน HashMap หรือค้นหาใน HashMap เพื่อที่จะได้เจอคำศัพท์ที่ต้องการ เนื่องจากตัวเล็กและตัวใหญ่โปรแกรมมองว่าไม่เหมือนกัน

- **ผลการประเมินตนเอง**

ให้ตนเองอยู่ที่ระดับ 80 เพราะสามารถทำงานได้ด้วยตัวเองแต่ก็ไม่ทั้งหมดมีตรงจุดที่โปรแกรมเกิดปัญหาเพราะไม่ได้ทำให้ String ที่เพิ่มเข้าไปนั้นมีรูปแบบตัวพิมพ์เล็กหรือใหญ่เหมือนกันทั้งหมดจึงทำให้โปรแกรมหาไม่เจอคำศัพท์เมื่อพิมพ์ตัวอักษรแบบตัวใหญ่เข้าไปเพราะใน HashMap นั้นเป็นตัวเล็กเลยเสียเวลาในการแก้จุดนี้ไปพอสมควรเพราะไม่ได้คิดถึง case นี้มาก่อนเนื่องจาก TreeSet และ ArrayList ไม่มีกรณีนี้เกิดขึ้นแต่ก็สามารถแก้ปัญหาได้และทำให้โปรแกรมรันได้แบบสมบูรณ์