Assignment 6 Array List

- ศึกษาการใช้คลาส List สำเร็จรูปที่มีอยู่ใน JAVA เช่น ArrayList
- เขียนโปรแกรมภาษาจาวา อ่านไฟล์ชื่อ "UTF8_Lexitron.csv"
  - ข้อมูลในแต่ละบรรทัดที่อ่านได้ประกอบด้วย คีย์เวิร์ด ความหมาย และชนิดของคำ
  - ดึงข้อมูลแต่ละบรรทัดมาสร้างพจนานุกรมแปลอังกฤษเป็น-ไทย
  - ข้อมูลบางบรรทัดจะมีคีย์เวิร์ดซ้ำกัน แต่ต่างกันที่ความหมาย หรือชนิดของคำ
  - ข้อมูลบางบรรทัดจะซ้ำกัน (ต่างกันที่จำนวนเว้นวรรคที่ปรากฏอยู่ในความหมาย)
  - ใช้คำสั่งที่มีอยู่ในจาวาได้ทุกคำสั่ง
- ขั้นตอนการการทำงานของโปรแกรม
  - อ่านข้อมูลจากไฟล์ นำไปเก็บในอาร์เรย์
  - แสดงผล จำนวนข้อมูลที่อ่านได้จากไฟล์ (74233)
  - เรียงลำดับข้อมูล
  - เปรียบเทียบเปรียบเทียบข้อมูลที่อยู่ติดกัน ถ้าเหมือนกันให้ลบทิ้ง (252)
    - แสดงจำนวนคำที่เหลือ (73981)
  - เปรียบเทียบ/นับ ข้อมูล(คีย์เวิร์ด) ที่อยู่ติดกัน จำตัวที่มีความหมายซ้ำมากที่สุด
    - แสดงผลตัวที่มีความหมายมากที่สุด (get off = 35 ตัว)
  - วนรอบ ตั้งคำถามหาคำศัพท์ แล้วค้นหา เพื่อแสดงผลคำศัพท์ทุกคำที่มี key word ตรงกัน
    - ต้อง trim และตัด space ตัวที่ไม่มีนัยสำคัญออก
    - แสดงผลคำศัพท์ที่มีคีย์เวิร์ดตรงกันทั้งหมด
      - สร้างฟังก์ชันค้นหาข้อมูลโดยใช้ binarySearch เมื่อเจอ ให้ค้นตัวที่อยู่ติดกันด้วย (ค้นย้อนไปตัวแรก แสดงผลถึงตัวสุดท้าย)
  - จบโปรแกรมด้วยการค้นคำว่า end ให้แสดงผลว่าจบโปรแกรม

Assignment 6 Test Case

- **Test Case วนรอบอ่านคำค้นจากคีย์บอร์ด แล้วแสดงผลข้อมูล (มีตำแหน่ง หรือ ลำดับ)**

Total Read 74233 records. // แสดงผลข้อมูลทั้งหมดที่อ่านได้
Total duplicate found 252 records. // แสดงผลจำนวนข้อมูลที่ซ้ำ (ตัดทิ้ง)
Total remaining size 73981 records. // แสดงผลจำนวนข้อมูลที่เหลือ
Maximun Meaning word get off have 35 meaning.
1) get off เริ่ม(PHRV)
.....
35) get off ออกจากรถ(PHRV)
Enter word : a // เจอ 1 ตัว (ตัวแรกที่ sort แล้ว)
found a 1 words at 0 - 0
1) A อักษรตัวแรกในภาษาอังกฤษ(N)
Enter word : zymurgy // เจอ 1 ตัว (ตัวสุดท้ายที่ sort แล้ว)
found zymurgy 1 words at 73980 - 73980
1) zymurgy การหมักสุรา(N)
Enter word : Gamine // เจอ 2 ตัว (ตัวแรกในไฟล์)
found Gamine 2 words at 23413 - 23414
1) gamine (เด็กหญิง) ซึ่งเล่นซุกซนแบบเด็กชาย(ADJ)
2) gamine เด็กหญิงที่ชอบเล่นซุกซนแบบเด็กชาย(N)
Enter word : CROON // เจอ 3 ตัว (ตัวสุดท้ายในไฟล์)
found CROON 3 words at 13684 - 13686
1) croon การฮัมเพลง(N)
2) croon ฮัมเพลง(VI)
3) croon ฮัมเพลง(VT)

Assignment 6 Test Case

Enter word : favorite // เจอ 4 ตัว (ตัวที่ถูกตัด)
found favorite 4 words at 20582 - 20585
1) favorite ซึ่งเป็นที่โปรดปราน(ADJ)
2) favorite คนโปรด(N)
3) favorite ความนิยมชมชอบ(N)
4) favorite ตัวเก็ง(N)

Enter word : □□□□acid □□□rain □□□ // มีเว้นวรรคหลายๆตัว เจอ 1 ตัว
found acid rain 1 words at 475 - 475
1) acid rain ฝนกรด(N)

Enter word : □□□□get □□□□off □□□ // มีเว้นวรรคหลายๆตัว เจอ 35 ตัว
found get off 35 words at 23963 - 23997
1) get off เริ่ม(PHRV)
.....
35) get off ออกจากรถ(PHRV)

Enter word : cpe // ไม่เจอ
cpe Not found

Enter word : end // เจอ 9 ตัว (ตัวจบโปรแกรม)
found end 9 words at 18763 - 18771
1) end เป้าหมาย(N)
.....
9) end ทำให้สิ้นสุด(VT)

End Program

Java Class : ArrayList<E>

- import java.util.ArrayList
  - ใช้สำหรับจัดการโครงสร้างข้อมูลแบบอาร์เรย์
  - ไม่สามารถอ้างถึงข้อมูลโดยตรงเหมือนกับการจัดอาร์เรย์ทั่วไป
  - มีคำสั่งทั่วไปในการจัดการข้อมูล แต่ต้องเรียกผ่าน method ที่กำหนดให้
- ตัวอย่างคำสั่ง method ที่เกี่ยวข้องกับคลาส ArrayList
  - การจองตัวแปรในคลาส List (ArrayList / LinkedList)
    - ArrayList<Dnode> dict = new ArrayList<Dnode> ();
  - การเพิ่มข้อมูลต่อท้าย dict.add(x);
  - การกำหนดค่าในตำแหน่งที่ i dict.set(int i, Dnode x);
  - การลบข้อมูลตำแหน่งที่ i dict.remove(int i);
  - การอ้างถึงข้อมูลในตำแหน่งที่ i x = dict.get(i);
  - นับจำนวนข้อมูล size() int i = dict.size();
  - การเรียงลำดับ Collections.sort(dict);
  - การค้นหา Collections.binarySearch(dict, key);
  - การวนรอบตั้งแต่ตัวแรก ถึงตัวสุดท้าย
    - for (int i = 0; i<dict.size(); i++) { .... ใช้งาน x = dict.get(i); ..... }
    - for (Dnode x : dict) { ..... ใช้งาน x จะเป็นข้อมูลตัวถัดไปในแต่ละรอบจนครบ ..... }

คำแนะนำ Assignment 6

- ใช้คลาส **ArrayList** ของจาวา เพื่อจัดการกับข้อมูลที่นำไปเก็บ
  - ข้อมูลคำศัพท์ 1 ตัวประกอบด้วย 3 ส่วนคือ คำศัพท์ , คำแปล และ ชนิดของคำ โดยใช้คำศัพท์เป็นคำค้น แล้วแสดงคำแปลและชนิดของคำออกมา
- สร้างคลาส เพื่อใช้จัดการข้อมูล 1 ตัว ที่ **implements Comparable** และ เมธอด **compareTo**

```
public class Dnode implements Comparable <Dnode> {
        String word;
        String mean;
        String type;
}
```

  - สร้างเมธอดชื่อ **compareTo** เพื่อเปรียบเทียบค่าคีย์แล้ว **return** ตัวเลข <0, 0 ,>0

```
public int compareTo(Dnode x) {//เปรียบเทียบสตริงในส่วนของ word
        return (int) this.word.compareToIgnoreCase(x.word);
}
```

  - สร้าง **constructor** ที่รับข้อมูลสตริง มาแบ่งเป็น 3 ส่วน โดยใช้คำสั่ง **split** แล้วนำไปเก็บในตัวแปรย่อยของคลาส //เพื่อใช้ในกรณีที่อ่านข้อมูลมา 1 ชุด แล้วส่งมาสร้าง object

```
public Dnode(String  buff) {
        String [] str = buff.split(",");
        word =str[0];
        mean=str[1];
        type=str[2];
}
```

  - อาจสร้างเมธอดอื่นๆ ที่กระทำกับข้อมูล 1 ตัว เช่น **toString** เพื่อแสดงผลข้อมูล(1 ตัว) ฯลฯ

คำแนะนำ Assignment 6

- สร้างเมธอดสำหรับเปรียบเทียบข้อมูลที่เหมือนกัน (เพื่อตัดคำซ้ำ)

```
boolean compareAll(Dnode x) {
    if (this.word.equalsIgnoreCase(x.word) && this.mean.equalsIgnoreCase(x.mean)
        &&  this.type.equalsIgnoreCase(x.type)
        return true;
    else return false;
}
```

- สร้างคลาส dictArray สำหรับรัน นำโครงสร้าง Dnode มาสร้าง ArrayList เพื่อนำไปใช้
  - ถ้าจอง ArrayList ใน main() ให้ส่ง dict ไปเป็นพารามิเตอร์ของ methods ต่างๆ
  - ถ้ากำหนด ArrayList เป็น static สามารถเรียกใช้ในทุก methods

    ArrayList <Dnode> dict = new ArrayList<Dnode>();
  - สร้างเมธอดสำหรับอ่านข้อมูลจากไฟล์กำหนด Encoding เป็น "UTF8" ทีละ 1 บรรทัด
    - ส่งบรรทัดที่อ่านได้ไปสร้าง // Dnode x = new Dnode(buff);
    - แล้วใส่เพิ่มในอาร์เรย์ลิสต์ // dict.add(x);

      // ข้อมูลที่อ่านได้จากไฟล์ **(74233)**
  - สร้างเมธอดสำหรับลบศัพท์คำที่ซ้ำกัน
    - วนรอบเปรียบเทียบข้อมูลตัวที่อยู่ติดกัน ถ้าเหมือนกัน ให้ลบทิ้ง **(252)**
  - สร้างเมธอดที่จำเป็นอื่นๆ เช่น นับคำศัพท์ที่ซ้ำมากที่สุด ค้นหาข้อมูล แสดงผลข้อมูล ฯลฯ
  - เรียงลำดับข้อมูลใช้ Collection.sort(dict)

ตัวอย่างคำสั่ง

- ตัวอย่างการใช้คำสั่ง
  - เรียงลำดับข้อมูลในอาร์เรย์โดยใช้ Collections.sort

```
Collections.sort(dict); // เรียกใช้ sort เพื่อเรียงลำดับข้อมูลในอาร์เรย์ลิสต์
```

  - ค้นหาข้อมูลในอาร์เรย์ด้วย binary search แล้วย้อนกลับไปตัวแรก

```
Dnode key = new Dnode (str); //สร้างโหนด key เพื่อค้นหาคำ str
int j = Collections.binarySearch(dict, key); // ค้นหาในข้อมูลทั้งหมด
        if (j > 0 && (dict.get(j-1).compartTo(key)==0) j-- ; //ถอยหลังไปถึงตัวแรก
```

  - เปรียบเทียบข้อมูลตัวที่อยู่ติดกัน ถ้าเจอลบออก

```
int remcount = 0;
for (int i=0; i<dict.size()-1; i++) )
   if (dict.get(i).compareAll(dict.get(i + 1))) //ต้องสร้าง compareAll ใน Dnode
     { dict.remove(i + 1);
       remcount++; } //นับจำนวนที่ลบไป
```

  - เมธอดสำหรับเปรียบเทียบข้อมูลที่เหมือนกัน (เพื่อตัดคำซ้ำ)

```
boolean compareAll(Dnode x) {//สร้างใน Dnode
    if (this.word.equalsIgnoreCase(x.word)
        && this.mean.equalsIgnoreCase(x.mean)
        &&  this.type.equalsIgnoreCase(x.type)
    return true;
    else return false; }
```

ตัวอย่างการอ่าน Text ไฟล์(UTF-8)

- กำหนด object ของไฟล์ โดยใช้ File InputStream เพื่ออ่านไฟล์ที่มี BOM
  - ไฟล์ UTF8 แต่ละตัวอักษรจะมีขนาด 2 bytes/char
    - จะมีรหัสเรียกว่า BOM มีขนาด 2 bytes (FEFF) อยู่ที่ต้นไฟล์ ต้องอ่านส่วนนี้ออกไปก่อนเพื่อไม่ให้กระทบข้อมูลจริงที่ต้องการ
    - การอ่านไฟล์ข้อมูลต้องกำหนดตัว Encoding ไว้เพื่อความถูกต้องในการแปลงเป็นตัวอักษร
  - ใช้คลาส FileInputStream คู่กับ InputStreamReader เพื่อระบุ Encoding

```
FileInputStream  fcsv = new FileInputStream("src\\UTF8_Lexitron.csv");
InputStreamReader utf = new InputStreamReader(fcsv,"UTF-8") ; //ระบุ encoding
```

- อ่านข้อมูลในไฟล์(Stream) โดยใช้ BufferedReader เพื่อให้อ่านข้อมูลทีละบรรทัดได้
  - อ่านข้อมูลผ่านคลาส BufferedReader ทีละบรรทัดด้วยคำสั่ง readLine()

```
BufferedReader fbuff = new BufferedReader(utf); // อ่านผ่าน BufferedReader
fbuff.read(); //อ่าน BOM ทิ้งไปก่อนครั้งแรก แล้วจึงเริ่มอ่านข้อมูลจริง
while  ( (str=fbuff.readLine()) != null ) { //วนรอบ อ่านข้อมูล ตรวจสอบว่าอ่านได้
   Dnode x = new Dnode(str); //นำข้อมูลที่อ่านได้ไปสร้างโหนดข้อมูล
   dict.add(x) ; // นำข้อมูล x ไปเพิ่มในอาร์เรย์ลิสต์ของ dict
} fsc.close() ; //ปิด
```

- ข้อมูลแต่ละบรรทัดในไฟล์ ประกอบด้วย 3 ส่วน คั่นด้วยเครื่องหมาย "," ใช้เป็นคำศัพท์ 1 คำ

```
gamine,(เด็กหญิง) ซึ่งเล่นซุกซนแบบเด็กชาย,ADJ
```

ตัวอย่างการอ่านไฟล์ UTF-8

```
public static int readLexitronFile(ArrayList<Dnode> data) {
        String buff = null;
        int count = 0;
        try {
        /* Set FileInputStream & Encoding */
            FileInputStream fr = new FileInputStream("src\\UTF8_Lexitron.csv");
            InputStreamReader csv = new InputStreamReader(fr, "UTF-8");
            BufferedReader fsc = new BufferedReader(csv);
            int BOM = fsc.read();
            while ((buff = fsc.readLine()) != null) {
                Dnode x = new Dnode(buff);
                data.add(x);
                count++;
            }
            fsc.close();
        } catch (FileNotFoundException e) {
                    System.out.println(e.getMessage());
                    System.out.println("File not found");
        } catch (Exception e) {
                    System.out.println("Error  " + e.getMessage());
                    System.out.println("Operation error");
        }
        return count;
}
```

File & Exception error

- Method ที่มีคำสั่งเกี่ยวกับไฟล์ จะมีคำสั่งที่สามารถทำให้เกิด Exception Error ขึ้นมาได้ ถ้าไม่มีการเขียนคำสั่งดักจับ try {..} catch {..} ไว้ในเมธอดนี้ จะไม่สามารถคอมไพล์ได้

```
void loadDataFile(String filename ) {
    int i = 0;
   try {
          File fr = new File(filename);
          Scanner sc = new Scanner(fr);
          System.out.format("Read file %s\n", filename);
          while (sc.hasNextLong()) { // อ่าน long เข้าในอาร์เรย์ data[]
              data[i++] = sc.nextLong();          }
          System.out.println(" Total  "+i+" records  read");
    } catch(FileNotFoundException e)
            {System.out.println("File not found ");} // แสดงเฉพาะกรณีเปิดไฟล์ไม่ได้
      catch(Exception e)
            {System.out.println("File read error ");} // กรณีอื่นๆทั้งหมด
}
```

throws Exception

- ถ้าไม่ต้องการเขียนตรวจสอบในเมธอดนี้ สามารถระบุให้ method ส่ง Exception ออกมา โดยใช้คำสั่ง **throws ExceptionClass** เขียนไว้หลังวงเล็บของพารามิเตอร์ เพื่อให้เมธอดที่เรียกใช้จัดการกับปัญหาของเมธอด นี้แทน (เมธอดที่เรียกใช้ก็จะต้องมีคำสั่ง **try {..} catch {..}** เพื่อตรวจสอบ หรือ throws ไปที่อื่น)

```
void loadDataFile(String filename) throws IOException {
    int i = 0;
    File   fr = new File(filename);
           Scanner sc = new Scanner(fr);
           System.out.printf("Read file %s\n", filename);
           while (sc.hasNextLong()) { // อ่าน long เข้าในอาร์เรย์ data[]
                data[i++] = sc.nextLong();
           }
           System.out.println(" Total  "+i+" records  read");
}
```

```
// Method ที่เรียกใช้ เมธอดที่มี throws exception จะต้องสร้าง try{}catch(){} ไว้ด้วย
 try {
          loadDataFile("FILE001.TXT");
      } catch(IOException e)
          {System.out.println("File not found");}
// ถ้า methods นี้ไม่สร้าง try{} catch(){}  ก็จะต้องทำ throws Exception ต่อออกไปเรื่อยๆ
```