Java Collections

- Class Collections เป็นการจัดเก็บข้อมูลที่เตรียมไว้ให้ใน Package java.util.* ซึ่งมีโครงสร้างการจัดเก็บให้เลือกใช้หลายวิธี เช่น List (ArrayList, LinkedList) Map(HashMap) Set(TreeSet)ฯลฯ
- ตัวอย่างคำสั่งที่ใช้กับคลาส

คำสั่ง	ArrayList (Array)	HashMap (Hashing)	TreeSet (Binary Search Tree)
เพิ่มข้อมูล obj	add(obj)	put(key, obj)	add(obj)
ค้นหา(boolean)	contains(obj)	containsKey(key) containsValue(obj)	contains(obj)
ค้นหาตำแหน่งที่เก็บ	indexOf(obj)	-	-
เรียกใช้ข้อมูล obj	get(index)	get(key)	subset(fromKey,b,toKey,b)
กำหนดค่าในตำแหน่ง	set(index, obj)	replace(key,obj)	-
ลบข้อมูล obj	remove(obj) remove(index)	remove(key)	remove(obj)
ขนาดข้อมูล(จำนวน)	size()	size()	size()

- วนรอบทุกตัว

```
for (int i = 0; i < dict.size() ; i++) { ... } //ArrayList
for (Element e : List) { ... } //List
```

- เรียงลำดับข้อมูลโดยใช้ implements Comparable <>

```
Collections.sort(List); //ArrayList
```

- ค้นหาข้อมูล โดยใช้ implements Comparable <>

```
j = Collections.binarySearch(List, E key); //ArrayList
TreeSet<E> s = subSet(key, true, key, true);  //TreeSet
```

Assignment 6 ArrayList

- **Topics** เรียนรู้การการใช้งาน Collections ในภาษาจาวา
- **Learning Outcomes**
  - มีแนวคิดในการใช้เรียกใช้ object ที่มีอยู่มาสร้างเป็น applications
- **โจทย์ปัญหา** สร้างโปรแกรมแปลคำศัพท์ (Dictionary) สำหรับการแปลอังกฤษเป็นไทย
  - ไฟล์ข้อมูลที่ใช้คือ "UTF8_Lexitron.csv" มีลักษณะดังนี้
    - ใช้ Encoding เป็น "UTF8" หรือ "UTF-8"
    - มี BOM (Byte order Marks) คือ "FEFF" (ต้องอ่านทิ้งก่อนด้วย .read แล้วจึงอ่านข้อมูลจริงไปใช้)
    - ข้อมูลในแต่ละบรรทัดที่อ่านได้ประกอบด้วย คีย์เวิร์ด ความหมาย และชนิดของคำ คั่นด้วย ","
    - คำศัพท์หรือคำแปลในแต่ละบรรทัดที่อ่านได้อาจมีซ้ำกัน (ต้องตัดที่ซ้ำกันทิ้ง)
- **ขั้นตอน**
  - เปิดไฟล์ "UTF8_Lexitron.csv" กำหนด Encoding แล้วอ่าน BOM ทิ้ง
  - อ่านข้อมูลทีละบรรทัดนำไปสร้าง obj (สร้าง constructor ไว้) แล้ว add ไปเก็บใน ArrayList (ต้อง trim และตัด space ตัวที่ไม่มีนัยสำคัญออกด้วย)
  - เรียงลำดับข้อมูล(Collections.sort) แล้วเปรียบเทียบข้อมูลตัวที่อยู่ติดกัน ถ้าเหมือนกันให้ลบทิ้ง
  - แสดงผลสรุปข้อมูลที่อ่านได้ (Test Case)
    - แสดงจำนวนข้อมูลที่อ่านได้จากไฟล์ (74233)
    - แสดงจำนวนคำที่ตัดทิ้ง (252)
    - แสดงจำนวนคำที่เหลือ (73981)
    - เปรียบเทียบ/นับ ข้อมูล(คีย์เวิร์ด) ที่อยู่ติดกัน แสดงผลตัวที่มีความหมายมากที่สุด (get off = 35 ตัว)
  - วนรอบ ตั้งคำถามหาคำศัพท์ แล้วค้นหา เพื่อแสดงผลคำศัพท์ทุกคำที่มี key word ตรงกัน
    - แสดงผลคำศัพท์ที่มีคีย์เวิร์ดตรงกันทั้งหมด
      - สร้างฟังก์ชันค้นหาข้อมูลจากคีย์ที่ป้อน โดยเรียกใช้ binarySearch
      - เนื่องจากมีคำศัพท์ที่มีคีย์ซ้ำกันหลายตัว (คนละความหมาย) การใช้ binarySearch อาจค้นเจอในตำแหน่งที่ไม่ใช่ตัวแรกของคำนั้น จึงต้องเปรียบเทียบถอยหลังกับตัวที่อยู่ติดกันเพื่อหาตัวแรก
      - แสดงผลคำศัพท์ที่เจอตัวแรก ต่อเนื่องจนถึงตัวสุดท้ายที่มีคีย์เวิร์ดเดียวกัน
  - จบโปรแกรมด้วยการค้นคำว่า end ให้แสดงผลว่าจบโปรแกรม

Array List Test Case

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

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

Array List Test Case

Enter token : favorite // เจอ 4 ตัว (ตัวที่ถูกตัด)
found favorite 4 tokens at 20582 - 20585
1) favorite ซึ่งเป็นที่โปรดปราน(ADJ)
2) favorite คนโปรด(N)
3) favorite ความนิยมชมชอบ(N)
4) favorite ตัวเก็ง(N)
Enter token : □□□□ **acid** □□□ **rain** □□□ // มีเว้นวรรคหลายๆตัว เจอ 1 ตัว
found acid rain 1 tokens at 475 - 475
1) acid rain ฝนกรด(N)
Enter token : □□□□ **get** □□□□ **off** □□□// มีเว้นวรรคหลายๆตัว เจอ 35 ตัว
found get off 35 tokens at 23963 - 23997
1) get off เริ่ม(PHRV)
.....
35) get off ออกจากรถ(PHRV)
Enter token : cpe // ไม่เจอ
cpe Not found
Enter token : end // เจอ 9 ตัว (ตัวจบโปรแกรม)
found end 9 tokens at 18763 - 18771
1) end เป้าหมาย(N)
.....
9) end ทำให้สิ้นสุด(VT)
End Program

Class ArrayList

- คำแนะนำการสร้างข้อมูล
  - สร้างคลาสสำหรับจัดการข้อมูล 1 ตัว ประกอบด้วย คีย์เวิร์ด, คำแปล, ชนิด (Dnode)
    - implements Comparable <Dnode> ใช้เมท็อด compareTo กับคีย์เวิร์ด
  - สร้างเมธอดสำหรับอ่านข้อมูลจากไฟล์กำหนด Encoding เป็น "UTF8" ทีละ 1 บรรทัด
    - ส่งบรรทัดที่อ่านได้ไปสร้าง Dnode (แยกคำใน Dnode ด้วย , แล้วนำไปเก็บ)
  - นำข้อมูลที่สร้างแล้ว ใส่เพิ่มในอาร์เรย์ลิสต์
- ตัวอย่างคำสั่งที่เกี่ยวข้องกับคลาส ArrayList
  - import java.util.ArrayList
  - การจองตัวแปรในคลาส ArrayList (ArrayList/ LinkedList)
    - ArrayList<Dnode> dict = new ArrayList<Dnode> ();
  - การเพิ่มข้อมูลต่อท้าย dict.add(x); //เทียบได้กับ dict[size++] = new Dnode(x);
  - การกำหนดค่าในตำแหน่งที่ i dict.set(i,x); // เทียบได้กับ dict[i] = x;
  - การลบข้อมูลตำแหน่งที่ i dict.remove(i); //for(j=i;j<count-1;j++) dict[j]=dict[j+1]; count--;
  - การอ้างถึงข้อมูลในตำแหน่งที่ i x = dict.get(i); // x = dict[i];
  - นับจำนวนข้อมูล size() i = dict.size(); // i=count;
  - การเรียงลำดับ Collections.sort(dict); // Array.sort(dict);
  - การค้นหา Collections.binarySearch(dict, obj); // Array.binarySearch(dict,obj);
  - การวนรอบตั้งแต่ตัวแรก ถึงตัวสุดท้าย
    - for (int i = 0; i<dict.size(); i++) { ....ใช้งาน x = dict.get(i); ..... }
    - for (Dnode x : dict) { ..... ใช้งาน instance x ..... }

ตัวอย่างการสร้างคลาส ข้อมูล

- สร้างคลาสเพื่อใช้จัดการข้อมูล 1 ตัว ที่ implements Comparable และ เมธอด compareTo เพื่อให้สามารถนำไปใช้กับ Arrays.sort() และ Arrays.binarySearch()

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

- สร้างเมท็อดชื่อ compareTo เพื่อเปรียบเทียบค่าคีย์แล้ว return ตัวเลข <0, 0 ,>0

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

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

```
public Dnode() {// สำหรับโหนดเปล่า new Dnode();
        word ="";
        mean = "";
        type = "";
}
public Dnode(String  buff) {//ส่งข้อมูล 1 บรรทัดมาสร้างโหนด new Dnode(buff);
        buff = buff.trim().replaceAll("\\s+","  ");
        String [] str = buff.split(",");
        word =str[0];
        mean=str[1];
        type=str[2];
}
```

```
while ((str = fbuff.readLine()) != null) {
        Dnode x = new Dnode(str);
        dict.add(x);
        count++;
    }
```

- สร้างเมท็อดอื่นๆ ที่กระทำกับข้อมูล 1 ตัว เช่น แสดงผลข้อมูล(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;
}
```

```
if (dict.get(i).compareAll(dict.get(i + 1))) {
    dict.remove(i + 1);
    j++; } //นับจำนวนที่ลบ
```

- สร้างคลาส dictArray สำหรับรัน นำโครงสร้าง Dnode มาสร้าง ArrayList เพื่อนำไปใช้
  - ถ้าจอง ArrayList ใน main() ให้ส่ง dict ไปเป็นพารามิเตอร์ของ methods ต่างๆ
    `void methodName(ArrayList<Dnode> dict, int count)`
  - ถ้ากำหนด ArrayList เป็น static สามารถเรียกใช้ในทุก methods
    `static 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)

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
   count++; //นับจำนวนบรรทัดที่อ่านได้
} fsc.close() ; //ปิด
```

  - ต้องมีการใช้ try { } catch ( ) { } เพื่อจัดการความผิดพลาด
- ข้อมูลแต่ละบรรทัดในไฟล์ ประกอบด้วย 3 ส่วน คั่นด้วยเครื่องหมาย "," ใช้เป็นคำศัพท์ 1 คำ

gamine,(เด็กหญิง) ซึ่งเล่นซุกซนแบบเด็กชาย,ADJ