

Project Array Part I

COS2103

Data structure and Algorithm

1 / 2568

Present to

Asst.Prof.DR.Chouvalit Khancome

By Krittaya Tantichaiyakul

6705500269

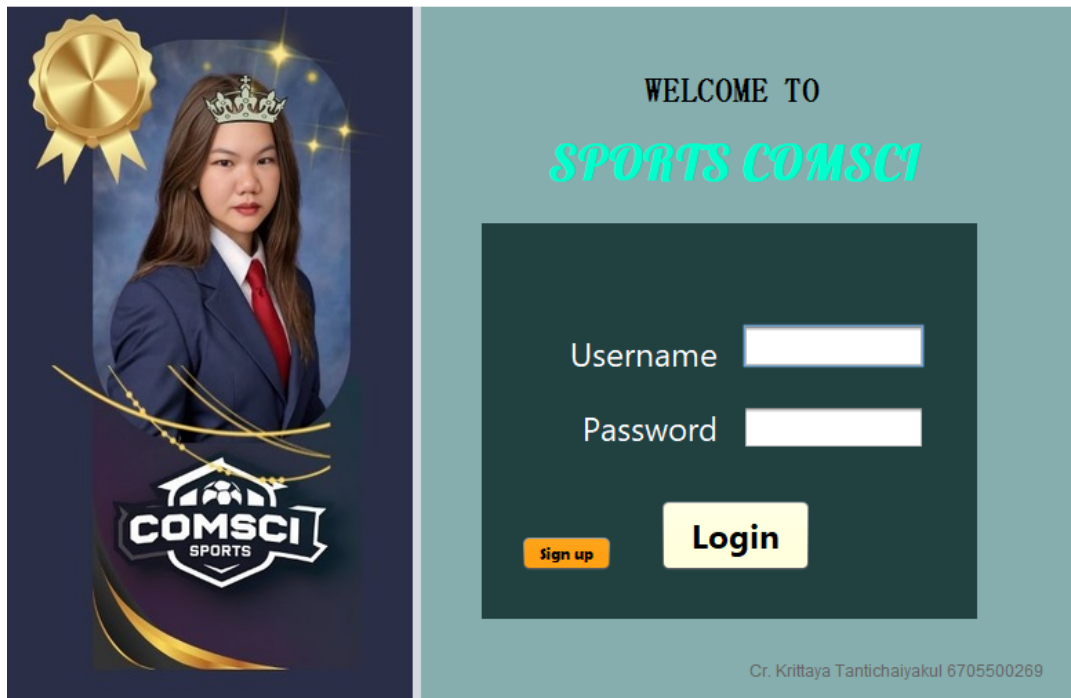
หัวข้อ

อุปกรณ์กีฬา



Program Interface Preview

1. Login page - Login



2. Category of array - Menu



3. list of products – Manage

DATA STRUCTOR AND ALGORITHM

Delete Insert

search code :

Host :
Krittaya001

Delete code

Status : delete complete!

Code : 10841

Name : Lacoste speed turbo

Price : 3500.0

Type : 1

Delete

No.	Code	Name	Price	Type	Address
1	10233	Adidas sne...	4600.0	1	/sc/sneake...
2	10472	Nike sneaker	4200.0	1	/sc/sneake...
3	10785	Skecher sn...	2500.0	1	/sc/ske.jpg
4	20096	Umbro Glove	800.0	2	/sc/ub.jpg
5	20504	Nike footba...	1500.0	2	/sc/shirtNik...
6	30319	Ping stick g...	1300.0	3	/sc/golfstic...
7	30873	Anyday golf ...	5000.0	3	/sc/golfbag...
8	40152	Yonex strike	2400.0	4	/sc/yon.png
9	40197	Kawasaki s...	1700.0	4	/sc/kawa.jpg
10	40601	Victor shuttl...	200.0	4	/sc/shuttle.j...
11	50290	Puma footb...	800.0	5	/sc/puma.jpg
12	50475	Mikasa voll...	1200.0	5	/sc/volley.jpg
13	50778	Wilson bas...	1500.0	5	/sc/wilson.j...

Stock

Storage : 15

Amount : 13

Present to Asst.Prof.DR.Chouvalit Khancome
by.. Krittaya Tantichaiyakul 6705500269

4. ไฟล์กลางจัดเก็บอาร์เรย์ – DataStore

```

11 public class DataStore {
12     public static int n=13;
13     final int UB=15;
14     int LB=0;
15     public static class Products {
16         public int code;
17         public String name;
18         public int type;
19         public float price;
20         public String pic;
21     }
22     public void setAll(int a, String b, int c, float d, String e){
23         code = a;
24         name = b;
25         type = c;
26         price = d;
27         pic = e;
28     }
29     static int getN(){
30         return n;
31     }
32 }
33
34
35 public static Products[] list = new Products[15];
36
37 static {
38     for (int i = 0; i < list.length; i++) {
39         list[i] = new Products();
40     }
41 }
42
43 // กำหนดข้อมูลตั้งแต่แรก
44 list[0].setAll(10233,"Adidas sneaker",1,4600,"/sc/sneakerAdi.jpg");
45 list[1].setAll(10472,"Nike sneaker",1,4200,"/sc/sneakerNike.jpg");
46 list[2].setAll(10785,"Skecher sneaker",1,2500,"/sc/ske.jpg");
47 list[3].setAll(20096,"Umbro Glove",2,800,"/sc/ub.jpg");
48 list[4].setAll(20504,"Nike football shirt",2,1500,"/sc/shirtNike.jpg");
49 list[5].setAll(30319,"Ping stick golf",3,1300,"/sc/golfstick.jpg");
50 list[6].setAll(30873,"Anyday golf bag",3,5000,"/sc/golfbag.jpg");
51 list[7].setAll(40152,"Yonex strike",4,2400,"/sc/yon.png");
52 list[8].setAll(40197,"Kawasaki strike",4,1700,"/sc/kawa.jpg");
53 list[9].setAll(40601,"Victor shuttlecock",4,200,"/sc/shuttle.jpg");
54 list[10].setAll(50290,"Puma football",5,800,"/sc/puma.jpg");
55 list[11].setAll(50475,"Mikasa volleyball",5,1200,"/sc/volley.jpg");
56 list[12].setAll(50778,"Wilson basketball",5,1500,"/sc/wilson.jpg");
57
58 }

```

การทำงานของอัลกอริทึม

1. สร้างไฟล์ DataStore เป็นศูนย์กลางสำหรับจัดเก็บอาร์เรย์ หากไฟล์อื่นต้องการเข้าถึงอาร์เรย์ จำเป็นต้องใช้ผ่านไฟล์นี้ เนื่องจากชุดข้อมูลที่อาจารย์กำหนดมี data type ที่แตกต่างกัน จึงต้องสร้าง Class Products เพื่อจัดเก็บ Attribute ทั้ง 5 สร้างอาร์เรย์ list ขึ้นมาขนาด 15 ช่อง พร้อมกำหนดค่าข้อมูล 13 ช่องคือ 0-12 และสร้างตัวแปร n สำหรับเก็บจำนวนสมาชิกในอาร์เรย์แบบ static โดยกำหนดให้ค่าเริ่มต้นเท่ากับจำนวนที่ใส่ข้อมูลไปคือ 13 ส่วน index ตำแหน่งที่ 13-14 ยังเป็นช่องว่าง เพื่อให้เกิดความหลากหลายในการเล่นกับอัลกอริทึมค่ะ

```
11 public class DataStore {
12     public static int n=13;
13     final int UB=15;
14     int LB=0;
15     public static class Products {
16         public int code;
17         public String name;
18         public int type;
19         public float price;
20         public String pic;
21     }
22     public void setAll(int a, String b, int c, float d, String e){
23         code = a;
24         name = b;
25         type = c;
26         price = d;
27         pic = e;
28     }
29     static int getN(){
30         return n;
31     }
32 }
33
```

Handwritten annotations for DataStore:
- Line 12: *จน. สิ้นสุด*
- Line 13: *Max*
- Line 14: *Start*
- Line 15-20: *attribute* (with arrows pointing to the Products class attributes)
- Line 29-30: *สามารถเรียก DataStore ได้* (with arrows pointing to the static methods)

```
35 public static Products[] list = new Products[15];
36
37 static {
38     for (int i = 0; i < list.length; i++) {
39         list[i] = new Products();
40     }
41 }
42
43 // กำหนดข้อมูลลงแต่ละช่อง
44 list[0].setAll(10233,"Adidas sneaker",1,4600,"/sc/sneakerAdi.jpg");
45 list[1].setAll(10472,"Nike sneaker",1,4200,"/sc/sneakerNike.jpg");
46 list[2].setAll(10785,"Skecher sneaker",1,2500,"/sc/ske.jpg");
47 list[3].setAll(20096,"Umbro Glove",2,800,"/sc/ub.jpg");
48 list[4].setAll(20504,"Nike football shirt",2,1500,"/sc/shirtNike.jpg");
49 list[5].setAll(30319,"Ping stick golf",3,1300,"/sc/golfstick.jpg");
50 list[6].setAll(30873,"Anyday golf bag",3,5000,"/sc/golfbag.jpg");
51 list[7].setAll(40152,"Yonex strike",4,2400,"/sc/yon.png");
52 list[8].setAll(40197,"Kawasaki strike",4,1700,"/sc/kawa.jpg");
53 list[9].setAll(40601,"Victor shuttlecock",4,200,"/sc/shuttle.jpg");
54 list[10].setAll(50290,"Puma football",5,800,"/sc/puma.jpg");
55 list[11].setAll(50475,"Mikasa volleyball",5,1200,"/sc/volley.jpg");
56 list[12].setAll(50778,"Wilson basketball",5,1500,"/sc/wilson.jpg");
57 }
58
```

Handwritten annotations for Products list:
- Line 35: *หรือสร้าง array object ขึ้นมาด้วย new*
- Line 40: *a b c d e* (with arrows pointing to the parameters in the setAll calls)
- Line 57: *list[0] and list[14] = Empty*

2. เริ่มต้นระบบจะพามาที่หน้า Login ท่านสามารถใส่ username และ password เป็น 1234 ได้เลย ซึ่งจริงๆแล้วหนูมีภาพในหัวอยู่แล้วว่า สามารถตั้งให้ผู้ใช้ sign in ใหม่ได้ โดยการจัดเก็บรหัสผ่านใหม่ไว้ที่อาร์เรย์ชุดหนึ่ง แล้วจึงตรวจสอบที่ปุ่ม Login หาก string ตรงกันกับชุดข้อมูล ก็จะอนุญาตให้เข้าหน้าถัดไปได้ แต่ถ้าทำแบบนั้นก็จะเป็น user ไม่ใช่เจ้าของที่จะไปสามารถจัดการชุดข้อมูลข้างในได้ เลขขอเก็บไว้ทำในโปรเจกต์หลังๆค่ะ 😊

WELCOME TO

SPORTS COMSCI

TextField1

Username

Password

Login

sign up

TextField2

Button1

Cr. Kittaya Tantichaiyakul 6705100269

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    int user=Integer.parseInt(jTextField1.getText());  
    int pass=Integer.parseInt(jTextField2.getText());
```

ทั้งค่า true

```
    if (user==1234 && pass==1234){  
        System.out.println("Login complete!");  
        JOptionPane.showMessageDialog(this, "ล็อกอินสำเร็จ!");  
        Menu f2 = new Menu(); // สร้างหน้าต่างใหม่  
        f2.setVisible(true); // แสดงหน้าต่างใหม่  
        this.dispose();  
    } else{
```

คำสั่งไป
หน้าต่างไป

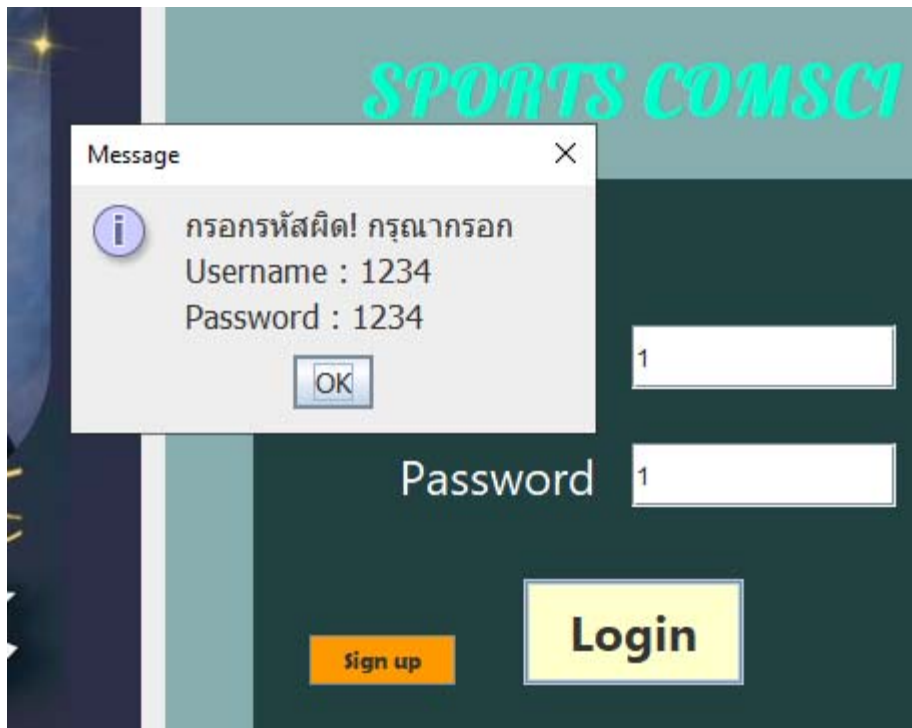
```
        JOptionPane.showMessageDialog(this, "กรอกรหัสผิด! กรุณากรอก\nUsername : 1234\nPassword : 1234"
```

```
    }  
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
```

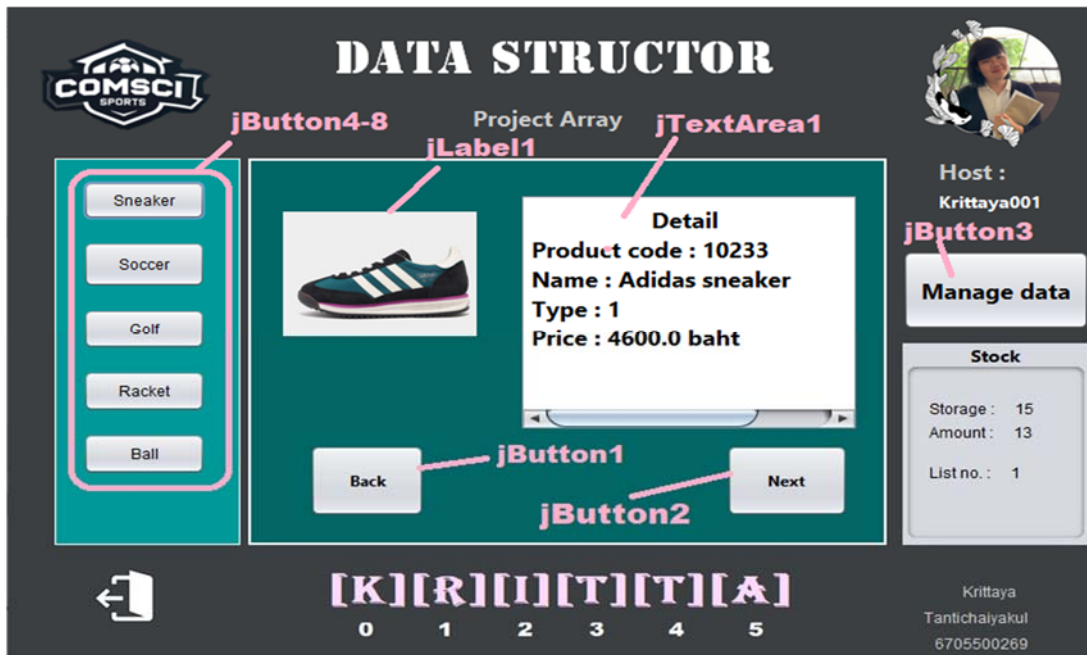
```
    UIManager.put("OptionPane.messageFont", new Font("Tahoma", Font.PLAIN, 16));  
    UIManager.put("OptionPane.buttonFont", new Font("Tahoma", Font.PLAIN, 14));  
    JOptionPane.showMessageDialog(this, "ขอภัยค่ะ ยังเป็นมือใหม่ รบกวนท่านกรอก\nUsername : 1234\nPass
```

popup
ข้อความ

หลังจากที่กด Login จะเข้าฟังก์ชัน jButton1 ซึ่งจะดึงข้อมูลจาก jTextField1 และ jTextField2 แปลงมาเป็น int แล้วเข้าคำสั่ง if ว่าทั้ง 2 texts นั้นเท่ากับ 1234 หรือไม่ ถ้าตรงก็อนุญาตให้เข้าหน้าถัดไป แต่ถ้ายังไม่ตรง ระบบก็จะ popup ขึ้นข้อความแสดงยูสเซอร์และรหัสผ่านค่ะ



3. หน้า Menu จะเป็นหน้าแสดงผลข้อมูลอาร์เรย์ที่ละรายการพร้อมรูปภาพประกอบ จะมีตัวแปรที่ชื่อ current ทำหน้าที่เก็บตำแหน่งปัจจุบันที่แสดงผล ดังนั้น JTextArea1 จะแสดงรายละเอียดข้อมูลและ JLabel1 จะแสดงรูปภาพของอาร์เรย์ตำแหน่งที่ current แบบ realtime กำหนดค่าเริ่มต้นให้ current = -1 เนื่องจากว่า หากตั้งเท่ากับ 0 เวลาผู้ใช้เข้ามาแล้วกดปุ่ม next เลย จะทำให้ค่า current+1 หรือเท่ากับตำแหน่งที่ 1 ทันที จะทำให้ข้ามข้อมูลตัวแรกไป



```

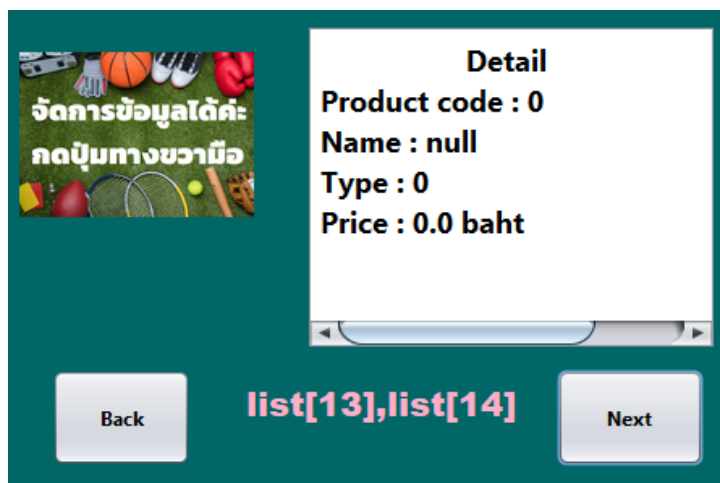
564 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
565     if (current < (DataStore.n-1)) {
566         current++;
567         jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource(DataStore.list[current].pic)));
568     } else if (current < (LB-1)) {
569         current++;
570         jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/sc/home.png")));
571     } else {
572         JOptionPane.showMessageDialog(this, "Overflow!");
573         System.out.println("Overflow!");
574     }
575     jLabel19.setText(""+(current+1));
576     jTextArea1.setText("
Detail"
+
"\nProduct code : "+DataStore.list[current].code+
"\nName : "+DataStore.list[current].name+
"\nType : "+DataStore.list[current].type+
"\nPrice : "+DataStore.list[current].price+" baht");
579
580
581
582
583
584 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
585     if (current > DataStore.n) {
586         current--;
587         jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/sc/home.png")));
588     } else if (current > LB) {
589         current--;
590         jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource(DataStore.list[current].pic)));
591     } else {
592         JOptionPane.showMessageDialog(this, "Underflow!");
593         System.out.println("Underflow!");
594     }
595

```

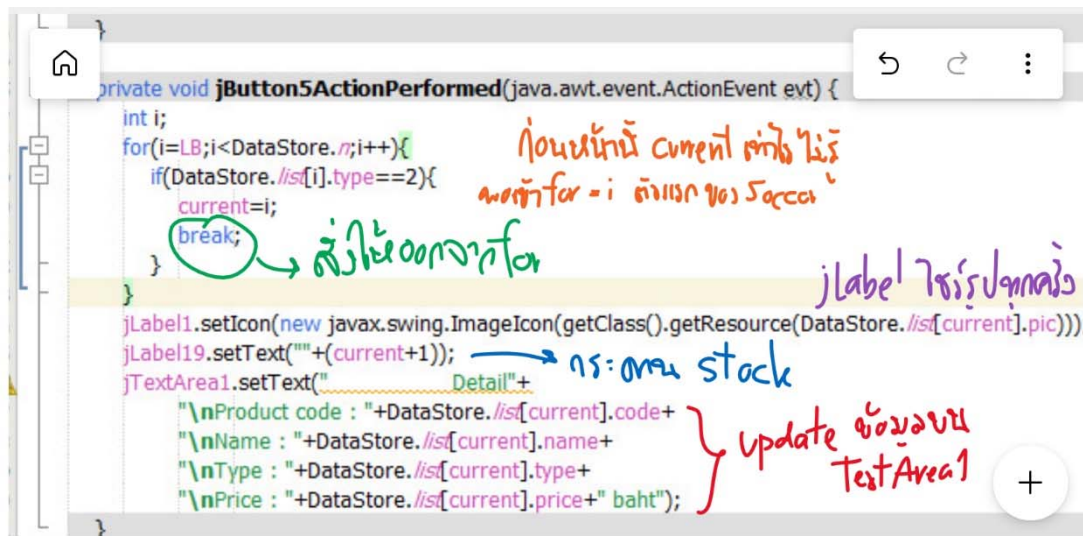
Handwritten annotations:

- LB - UB**: Labelled on the left side of the code block.
- add default**: Points to the line where the icon is set to "/sc/home.png" when the current index is out of bounds.
- set icon**: Points to the line where the icon is set to the selected item's picture.
- show in text area**: Points to the line where the item details are formatted and set to jTextArea1.
- 0...14**: A green box highlighting the range of indices in the code.

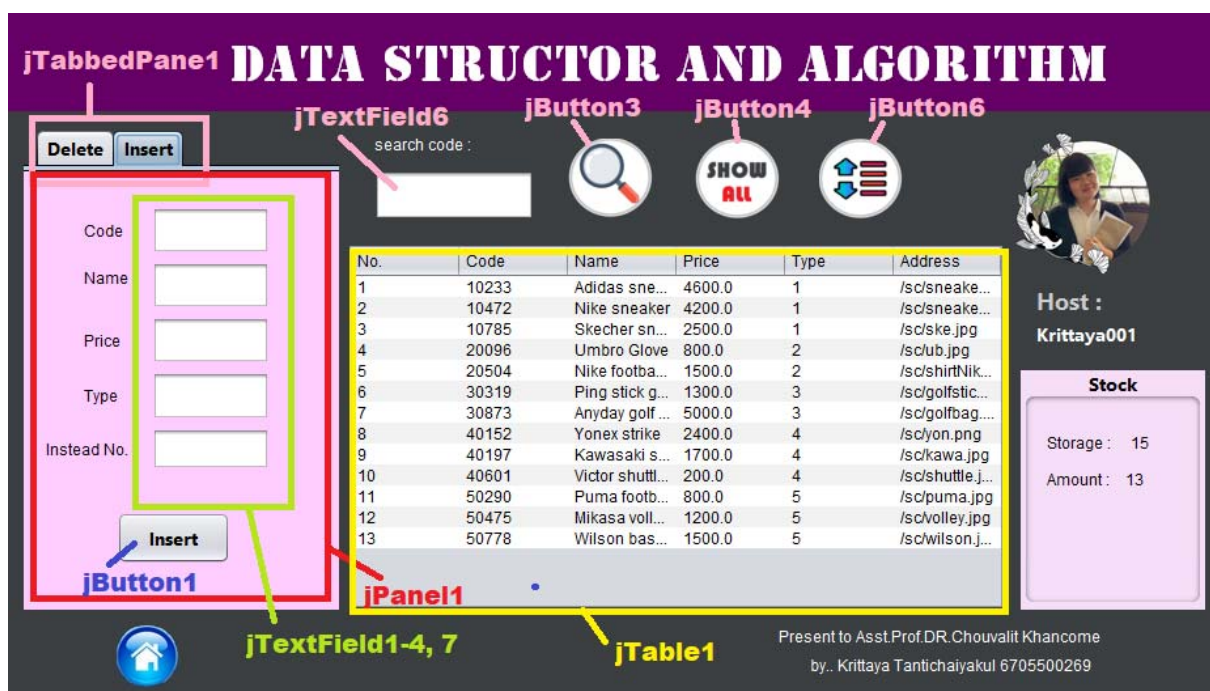
กำหนดให้ปุ่ม next และ back แสดงข้อมูลถัดไปและก่อนหน้า โดยกำหนดสเกลไว้ที่ 0-UB โดยการใช้ if else หากอยู่ในเงื่อนไข จึงอนุญาตให้ + - current ได้ จะทำให้เวลากด next back จะไม่เกินขอบเขตที่มีอยู่ แล้วจึงแสดงผลผ่าน TextArea1 และ JLabel1 อีกครั้ง และเนื่องจากว่าภายในกล่องอาร์เรย์เรายังมีข้อมูลที่ยังไม่ได้ใส่ข้อมูล ก็คือ index ที่ 13 14 เวลากด next มาถึงตำแหน่งนี้จะเกิด error เนื่องจากปกติแล้ว JLabel1 จะดึงรูปภาพจาก address ที่อยู่ในแอททริบิวต์ string pic ทุกครั้ง แต่ในตำแหน่งที่ 13 14 เป็นช่องว่างไม่มี address กด next จะขึ้นข้อความแดง ดังนั้นเราควรหารูป default ที่ใช้ไปรูปนั้นๆ เวลาไม่มีข้อมูลด้วย



แถบปุ่ม Category ทางซ้ายมือ คือหมวดหมู่ของรายการสินค้า หากท่านกดที่ปุ่มหมวดใด ระบบจะพาท่านไปยังรายการแรกของหมวดนั้นๆ คล้ายๆสารบัญ ตัวอย่าง หมวด Soccer จะมี type คือ 2 เวลากด ระบบจะเข้าฟังก์ชัน jButton5 สร้างตัวแปร i ข้างใน ในการทองลูป เนื่องจากเราต้องการให้พาไปรายการแรกของหมวดนั้นๆ จะต้องเริ่มต้นที่ 0 ไปถึง n-1 หากเจอ type ที่ตรงกันตัวแรก ก็แค่กำหนด current เท่ากับ i ได้เลย พร้อมอัปเดตข้อมูลลง Area1 และรูปภาพลง JLabel1 ภายในฟังก์ชันนั้นๆ



4. การจัดการข้อมูล จำเป็นต้องเข้าไปยังหน้าไฟล์ Manage หรือทำการกดปุ่ม jButton3 จากหน้า Menu นั้นเอง หน้านี้จะสร้าง jTable1 ขนาด 15 ช่องตามขนาดของอาร์เรย์ ท่านจะเห็นตารางจะแสดงข้อมูลสินค้าทั้งหมด 13 ชิ้น ซึ่งต่างจากหน้า Menu ที่จะแสดงข้อมูลที่ละรายการ อีกทั้งตารางนี้มีการเช็คไว้ในทุกๆปุ่มหรือทุกๆฟังก์ชันว่า เวลาที่มีการจัดการข้อมูลในรูปแบบต่างๆ ตารางนี้ก็จะอัปเดตแบบ realtime ด้วย



4.1 Insert

การแทรกข้อมูลใหม่ลงไปในกลุ่มอาร์เรย์ ท่านจำเป็นต้องกรอกรายละเอียดลงบน jTextField1-4 และ 7 ก่อน ค่อยกดปุ่ม Insert ระบบจะเข้าฟังก์ชัน jButton1 เริ่มต้นหนูกำหนดตัวแปรขึ้นมา 2 ตัวเพื่อรับค่าที่จำเป็นสำหรับการแทรกข้อมูลก็คือ โค้ดสินค้าและตำแหน่งที่ต้องการแทรก (jTextField1 & jTextField7) นำมาแปลงจาก string เป็น int แล้วทำการลบค่าตำแหน่งลง 1 เพื่อนำมาใช้ในอาร์เรย์ หลังจากนั้นจะเข้าลูปรตรวจสอบว่า โค้ด(jTextField1) ที่รับค่ามานั้น มีเลขนี้อยู่ในอาร์เรย์อยู่แล้วหรือไม่ ถ้าตรงกันจะให้เด้ง popup แสดงข้อความว่า มีเลขโค้ดนี้อยู่แล้ว และ return เพื่อออกจากฟังก์ชัน แล้วให้ผู้ใช้ไปแก้ไขเปลี่ยนเลขอื่นแทน แต่ถ้าไม่มีเลขซ้ำ จะเข้าคำสั่ง if ถัดไปเพื่อตรวจสอบว่า ขณะนี้ชุดข้อมูลมากกว่าหรือเท่ากับ UB หรือไม่ ซึ่งเราเซตไว้ข้างบนแล้วให้เท่ากับ 15 หากเป็นจริงแสดงว่าช่องอาร์เรย์เต็มเรียบร้อยแล้วทำการ return ออก แต่ถ้ายังไม่เต็ม ก็จะเข้า if ที่ 2 และ 3 เพื่อตรวจสอบว่าตำแหน่งที่ผู้ใช้ส่งมา(jTextField7) นั้น น้อยกว่า 0 หรือมากกว่า UB ไหมตามลำดับ พุดง่าย ๆ ก็คือ อยู่ในขอบเขตของอาร์เรย์หรือไม่ไหม หากเป็นจริงก็จะเด้งข้อความ Underflow และ Overflow ตามลำดับ พร้อม return ออก หากผ่านประตู if ทั้ง 4 มาได้แล้ว นั้นหมายความว่า การแทรกข้อมูลตรงตามเงื่อนไขทุกอย่าง พร้อมทั้งพาเข้าลูปรย้อนหลัง การเข้าลูปรย้อนหลังคือการเริ่มจาก ตัวท้ายสุดมายังตัวที่กำหนด เนื่องจากว่าการแทรก คือการเพิ่มจำนวน เราต้องถอยตัวสุดท้ายออกไป 1 ช่อง และตัวที่เหลือก็เขยิบตามกันมา ดังนั้นตัวที่ใกล้ช่องว่างมากที่สุดก็คือ ตัวสุดท้ายนั่นเอง ข้อมูลที่อยู่หลังจาก insertIndex จะมีค่าตำแหน่ง +1 ทุกตัวและข้อมูลใหม่ที่ได้จาก jTextField2-4 ก็จะถูกเข้ามาจัดเก็บลงในอาร์เรย์ช่องที่ insertIndex แทนที่ หลังจากนั้นจึงอัปเดตค่า n+1 ตารางและกระดานสต็อคค่ะ

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int insertCode=Integer.parseInt(jTextField1.getText());
    int insertIndex=Integer.parseInt(jTextField7.getText());
    insertIndex--;
    for(int i=0;i<DataStore.n;i++){
        if(insertCode==DataStore.list[i].code){
            System.out.println("This code exist already");
            JOptionPane.showMessageDialog(this, "This code exist already!");
            return;
        }
    }
    if(DataStore.n>=UB){
        System.out.println("Array full!");
        JOptionPane.showMessageDialog(this, "Array full!");
        return;
    }
    if(insertIndex<0){
        System.out.println("Underflow!");
        JOptionPane.showMessageDialog(this, "Underflow!");
        return;
    }
    if(insertIndex>=(UB-1)){
        System.out.println("Overflow!");
        JOptionPane.showMessageDialog(this, "Overflow!");
        return;
    }
    for(int j=(DataStore.n-1);j>=insertIndex;j--){
        DataStore.Products temp = new DataStore.Products();
        temp.code = DataStore.list[j].code;
        temp.name = DataStore.list[j].name;
        temp.price = DataStore.list[j].price;
        temp.type = DataStore.list[j].type;
        temp.pic = DataStore.list[j].pic;
        DataStore.list[j+1]=temp;
    }

    DataStore.list[insertIndex].code=Integer.parseInt(jTextField1.getText());
    DataStore.list[insertIndex].name=jTextField2.getText();
    DataStore.list[insertIndex].price=Integer.parseInt(jTextField3.getText());
    DataStore.list[insertIndex].type=Integer.parseInt(jTextField4.getText());
    DataStore.list[insertIndex].pic="/sc/home.png";
    DataStore.n++;
    System.out.println("Status : insert complete!");
    jLabel21.setText("Amount : "+DataStore.n);
    loadTableData();
}

```

Handwritten Notes:

- max** (near UB)
- stack บั๊ว** (near Array full!)
- index** (near Overflow!)
- update n** (near DataStore.n++)
- table** (near loadTableData())
- ns: table** (near loadTableData())

Diagram 1: Array [1, 2, 3, 4, 4] with index 3 pointing to the second 4.

Diagram 2: Array [1, 2, 3] with index 2 pointing to the 3.

Diagram 3: Array [1, 2, 9, 3]

↑

Insert Function

4.2 Search

สำหรับการค้นหาข้อมูล ผู้ใช้จะกรอกค่าเพียง 1 ค่า คือ โค้ดสินค้า แล้วจึงทำการสร้างตัวแปร index เพื่อคอยรับค่าที่เรากลับ ที่ส่งไปยังฟังก์ชัน searchData พร้อมกับแนบอาทิวนั้นไป นั่นก็คือ โค้ดที่เราเพิ่งแปลงมาจาก jTextField6 ภายในฟังก์ชันนั้น เราสร้างตัวแปรเก็บผลลัพธ์ LOC = -1 และตัวท่อนอาร์เรย์ I=0 ปกติของอาจารย์จะรีเทิร์นกลับเป็น Boolean แต่ของหนูขออนุญาตดัดแปลงนิดนึงละ ขอส่งกลับเป็น int นะคะ 🙏 เพื่อจะส่ง index ที่ต้องการค้นหา กลับไปด้วย โดยที่ทำการวนลูปหา หากโค้ดสินค้าตรงกันที่ I ไค ก็กำหนดค่าให้ LOC เท่ากับ I ได้เลยและส่ง LOC กลับ แต่ถ้าไม่เจอ ก็ยังคงคอนเซ็ปเดิมคือ ส่งค่า LOC ที่ไม่ได้มีการเปลี่ยนแปลงค่า ซึ่งก็คือ -1 กลับไปนั่นเอง

```
public int searchData(int ITEM){
    int LOC=-1; if = -1
    int I=0;      = false
    while(I<DataStore.n){
        if(ITEM==DataStore.list[I].code){
            LOC=I; = Index ≠ false
            System.out.println("Found at index : "+I);
            return LOC;
        }else{
            I++;
        }
    }
    System.out.println("Not found");
    return LOC;
}
```

Annotations:

- Parameter = Code Products** (points to `int ITEM`)
- ตัวท่อน Loop** (points to `while(I<DataStore.n){`)
- เจอ! ส่ง index** (points to `return LOC;`)
- ยังไม่เจอ ส่ง -1** (points to `return LOC;` at the end)
- ยังหาค่าไม่เจอ ไม่ส่งค่าอะไรไป** (points to the `else` block)
- still false -1** (points to the final `return LOC;`)

หลังจากกลับมาจาก searchData ก็จะเข้า if ตรวจสอบสถานะว่าเจอหรือไม่ โดยที่ค่า LOC หาก < 0 หรือจะตั้ง $= -1$ ก็ได้เหมือนกัน หากใช่จะแจ้งเตือนและรีเทิร์นออก แต่ถ้าผ่าน if นี้ นั่นหมายความว่าเราจะได้อ่า LOC ที่เป็น index กลับมา และทำการเช็ตตารางให้โชว์แค่ข้อมูลในตำแหน่งที่ index เท่านั้น

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    int SearchCode= Integer.parseInt(jTextField6.getText());  
    int index=SearchData(SearchCode);  
    if(index<0){  
        JOptionPane.showMessageDialog(this, "not found!");  
        return;  
    }  
    DefaultTableModel model = (DefaultTableModel) jTable1.getModel();  
    model.setRowCount(0);  
    model.addRow(new Object[]{index+1,  
        DataStore.list[index].code,  
        DataStore.list[index].name,  
        DataStore.list[index].price,  
        DataStore.list[index].type,  
        DataStore.list[index].pic  
    });  
}
```

← แปลงค่าใน jTextField6
← ได้ค่า loc กลับมา
← ถ้าไม่เจอจะได้ -1 ถ้า if นี้
← Reset Table
← show ค่า , เลข = ตำแหน่งที่ค้นห

4.3 Delete

สำหรับการลบข้อมูล จะใช้ข้อมูลจากผู้ใช้คือ โค้ดสินค้า ใช้วิธีเดียวกันกับการ Search เลย เพราะเราจำเป็นต้องหาก่อนว่า โค้ดสินค้าที่ต้องการจะลบ ตรงกับชุดข้อมูลในอาเรย์ในตำแหน่งที่เท่าไร โดยการโยนค่าโค้ดสินค้านี้ลงบนฟังก์ชัน searchData

หลังจากกลับมาจาก searchData ก็จะเข้า if ตรวจสอบสถานะว่าเจอหรือไม่ เหมือนกับ ฟังก์ชันค้นหา แล้วก็ทำการสร้าง loop ให้เริ่มต้นตั้งแต่ index ที่ได้รับมา เราจะเอาช่องนั้นให้มีค่าเท่ากับอาเรย์ช่องถัดไป ซึ่งข้อมูลเดิมตำแหน่งที่ index จะหายไปแล้ว ทำวนไปเรื่อยๆ จนครบ loop ในตอนนี้ เราจะได้ข้อมูลอาเรย์ที่มีจำนวนเท่าเดิม แต่จะมีข้อมูลที่เหมือนกัน 2 ตัว อยู่ท้ายแถว เราจะต้องลบข้อมูลตัวสุดท้ายออก โดยการกำหนดให้ตัวสุดท้ายเท่ากับ null พร้อมทั้งอัปเดต n-1 ตารางและกระดานสต็อกด้วย

The image shows a screenshot of a Java IDE with code for a delete function. The code is as follows:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    int delCode = Integer.parseInt(jTextField5.getText());  
    int index = searchData(delCode);  
    if(index < 0){  
        System.out.println("Status : not found");  
        jLabel11.setText("Status : not found..");  
        return;  
    }  
    jLabel12.setText("Code : "+DataStore.list[index].code);  
    jLabel13.setText("Name : "+DataStore.list[index].name);  
    jLabel14.setText("Price : "+DataStore.list[index].price);  
    jLabel15.setText("Type : "+DataStore.list[index].type);  
    for(int i=index; i<(DataStore.n-1); i++){  
        DataStore.list[i] = DataStore.list[i+1];  
    }  
    DataStore.list[DataStore.n-1] = null;  
    DataStore.n--;  
    System.out.println("Status : delete complete");  
    jLabel11.setText("Status : delete complete!");  
    jLabel21.setText("Amount : "+DataStore.n);  
    loadTableData();  
}
```

Handwritten annotations in Thai provide a detailed explanation of the code:

- เปลี่ยนเป็น int ส่งไปหาคำสั่ง delete**: A note pointing to the `delCode` variable.
- ไม่ได้ค่า Loc กลับมา**: A note pointing to the `index` variable, indicating that if it's less than 0, the item was not found.
- ก็ไม่ได้ลบก็จะไม่ -1 กลับมาทำที่นี้**: A note pointing to the `return` statement, explaining that if the item is not found, the count `n` is not decremented.
- ได้ข้อมูลลบ Panel จำนวนตัวที่ลบจะลดลงตรงค่า n ไปแล้ว**: A note pointing to the `DataStore.n--` line, explaining that the panel count and the `n` value are updated.
- update ตาราง และ กระดาน**: A note pointing to the `loadTableData()` call, indicating that the table and board are updated.

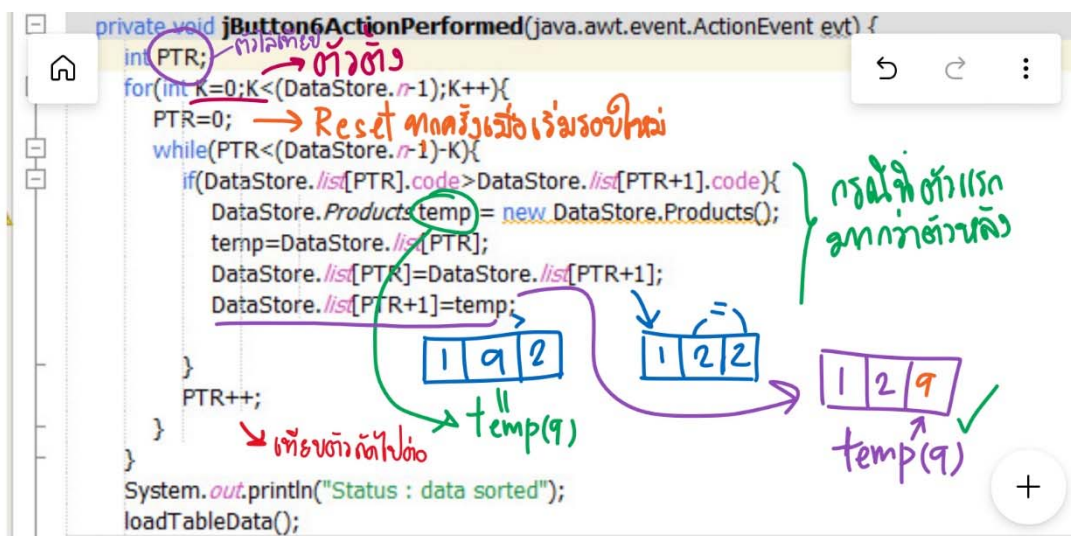
Diagram illustrating the array shifting process:

Initial array: `[1, 2, 3, 3]` (index 0 to 3). The element at index 2 (value 3) is the one to be deleted. The elements from index 3 onwards are shifted one position to the left.

Resulting array: `[1, 2, 3, null]`. The last element is now `null`, and the count `n` is decremented from 4 to 3.

4.4 Sorting

หากผู้ใช้มีการแทรกข้อมูลเข้าไปในอาร์เรย์ โดยข้อมูลไม่ได้เรียงกันถูกต้อง ท่านสามารถกด sort ได้ จะพาท่านไปยังฟังก์ชัน jButton6 อัลกอริทึมของอาจารย์จะเป็นการเรียงลำดับแบบ bubble sort โดยที่รอบแรกจะได้ตัวมากที่สุดอยู่ในลำดับสุดท้ายและไล่มาเรื่อยๆจนถึงค่าน้อยที่สุดตัวแรก เราจะสร้าง 2 ลูป เพราะเราจะต้องไล่เทียบ 2 ตัวแปร ตัวแปรแรกคือ K หรือตัวตั้งไว้ในลูปแรก เริ่มต้นที่ 0 และ PTR ตัวเทียบไว้ในลูปที่สอง คอยไล่สมาชิกทุกตัวมาเทียบกับตัวตั้ง K ในทุกๆรอบ ในแต่ละรอบจะกำหนดให้ PTR = 0 เสมอ เพื่อให้เริ่มวิ่งตั้งแต่ตัวแรกทุกครั้ง จนถึงตัวที่ $n-K$ สาเหตุที่ต้องลบ K ออกในลูปที่ 2 เนื่องจาก ในแต่ละรอบ เราจะได้ตัวมากที่สุดมาที่ท้ายแถวเรียบร้อยแล้ว ไม่ว่าจะเป็นรอบที่ 2 หรือ 3 หรือรอบที่ใดๆก็ตาม เราจะได้ตัวมากที่สุดจำนวน K ตัว(รอบ)มาที่ท้ายแถวของเรา ดังนั้น เราไม่จำเป็นต้องวนเทียบทุกอีลิเมนต์ในรอบต่อๆมาก็ได้ ซึ่งตรงนี้เป็นหัวใจสำคัญในการเขียนอัลกอริทึมที่ดี



ขอบคุณค่ะ

