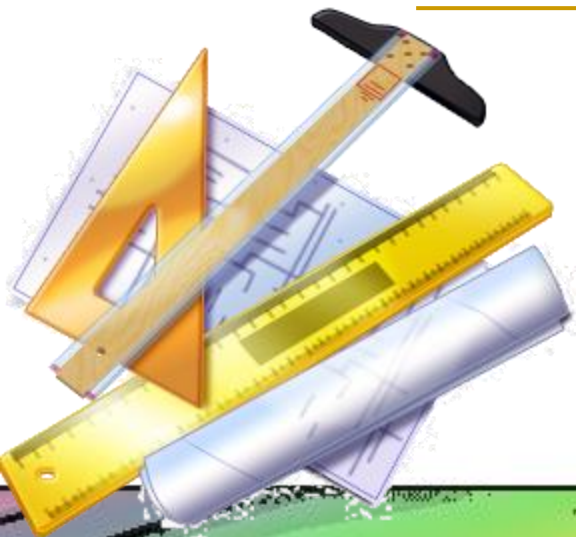


Array และ Array Lists

Lecture 7

Yaowadee Temtanapat

เยาวดี เต็มธนาภักดิ์



วัตถุประสงค์ของการเรียนในวันนี้

- เข้าใจแนวคิดเกี่ยวกับ Collection
- รู้จักการใช้ array และ array list เพื่อการเก็บกลุ่มของข้อมูล
- สามารถประกาศคลาสที่มี array และ array list เป็นสมาชิก
- อธิบายการสร้าง array 2 มิติ (2-dimensional) จากอาร์เรย์ของอาร์เรย์
- เรียนรู้เกี่ยวกับ algorithm พื้นฐานเพื่อทำงานกับกลุ่มของข้อมูล เช่นการค้นห การจัดการกับสมาชิกใน Collection
 - เข้าถึงหน่วยย่อยในกลุ่มข้อมูลที่ถูกเก็บใน array, array lists
 - implement array เพื่อการขยาย โดยเก็บค่าเพียงบางส่วนก่อน
 - การส่งผ่าน array ไปยัง methods

แนวคิดเกี่ยวกับวัตถุ Collection

- *Collection* : วัตถุที่ใช้ในการเก็บกลุ่มของวัตถุ (Collection of objects)
เช่นกลุ่มนักศึกษา
- การดำเนินการพื้นฐานที่ควรมีใน Collection:
 - การเพิ่มวัตถุใน Collection
 - การลบวัตถุออกจาก Collection
 - การตรวจสอบว่ามีวัตถุที่ต้องการอยู่ใน Collection หรือไม่
 - การทำงานเฉพาะบางอย่างกับวัตถุใน Collection เช่นเพิ่มค่าทุกสมาชิกขึ้นอีก 1
หรือ "**for each** item in the collection, do ... "

ลักษณะของ Collection

■ ทั่วไป Collection เป็น Homogeneous Collection

- ทุกสมาชิกใน Collection ควรเป็นชนิดเดียวกัน

- ช่วยให้ทราบแน่นอนว่าสามารถทำอะไรกับวัตถุใน Collection นั้นได้บ้าง

■ ตัวอย่างของ Collection เช่น

- เซต (Set): ไม่เก็บสมาชิกซ้ำ (no duplication) และ ไม่มีลำดับ

- ลิสต์ (List): เก็บสมาชิกแบบมีลำดับ (Sequence) และมีขนาดจำกัด (finite) โดยอาจมีสมาชิกซ้ำกันได้

- ทำให้สามารถอ้างอิงสมาชิกด้วยตำแหน่งได้ เช่น สมาชิกตัวที่ 1, ตัวสุดท้ายและอื่น ๆ

- อาร์เรย์ (array): built-in ของภาษา ใช้เก็บสมาชิกชนิดเดียวกัน รู้จำนวนแน่นอน

Array: การใช้ array ในการเก็บข้อมูล

- Array เป็นที่เก็บกลุ่มของข้อมูลที่เป็นชนิดเดียวกัน แต่ละหน่วยย่อยของข้อมูลสามารถที่จะถูกเข้าถึงได้แยกจากกัน
- การสร้าง array: สร้างและกำหนดขนาดของ array โดย new
- Java Syntax:

```
new typeName[length]
```

- ตัวอย่างเช่น

```
new double[4];
```

- จุดมุ่งหมาย เพื่อสร้าง array ที่มีขนาดเท่ากับจำนวนที่กำหนด

พื้นฐานเกี่ยวกับ Array

■ Array ใน Java เป็น Object

- การประกาศ: `float[] commission;` หรือ `float commission[];`

- หากใช้ `float[] x, y, z;` หมายความว่า x, y, z เป็น array

- ใช้ new statement เพื่อสร้าง: `commission = new float[6];`

■ Index ของ array เริ่มต้นจาก 0 ถึง length - 1

■ ค่าเริ่มต้นของ array:

- เมื่อสร้าง array ทุกสมาชิกถูกให้ค่าเริ่มต้นด้วย *ค่าปริยาย*

- เช่น array ของ char จะถูก set ค่าสมาชิกเริ่มต้นเป็น null (`\u0000`) character

ข้อควรระวังเกี่ยวกับ array

- การใช้ array ก่อนที่จะมีการสร้าง เช่น

```
double [] data;  
data[0] = 1.5; // Error used before construct
```

- สามารถหาขนาดของ array ได้จาก constant *length*

- Error ถ้าใช้ array เกินขอบเขตของขนาดของ array

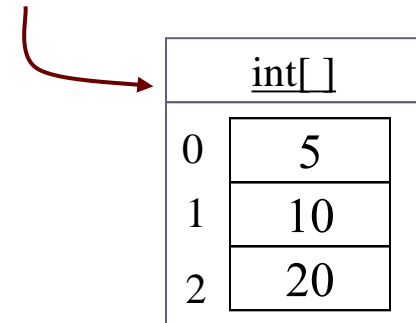
```
double [] data = new double[4];  
data[4] = 1.5; // Error array ขนาด = 4 มีช่วงที่เป็นได้คือตั้งแต่ 0 ถึง 3
```

Array ของ Primitive Data Types

■ ตัวอย่าง

```
int[] boxSize = new int[3];  
boxSize[0] = 5;  
boxSize[1] = 10;  
boxSize[2] = 20;
```

boxSize



int[]	
0	5
1	10
2	20

■ ใช้ *length* ในการบอกค่าขนาดของ array เช่น

- ❑ `boxSize.length;` // ในกรณีนี้ค่าที่ได้ คือ 3
- ❑ สังเกต `length` เป็นค่าคงที่ ไม่ใช่ method เหมือนอย่างใน `String.length();`

การวน loop เพื่อเข้าถึงสมาชิกในอาร์เรย์

■ Java Syntax: การวน loop ในแบบ for each

for (*FormalParameter* : *Expression*)
 Statements;

❑ ตัวอย่างเช่น

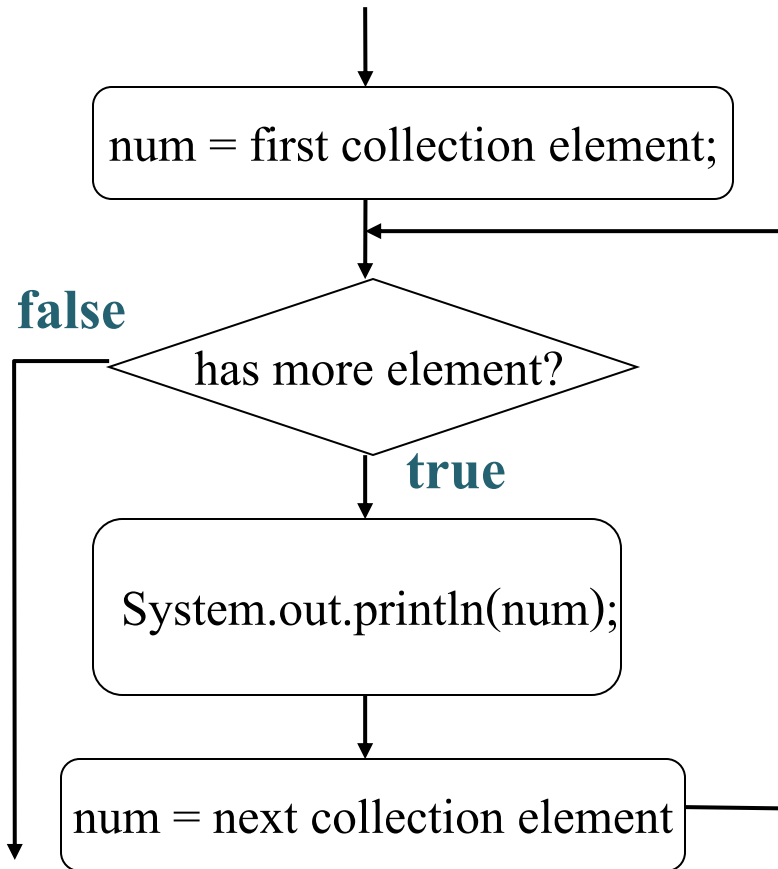
```
for (int num: boxSize) { ... }
```

❑ จุดมุ่งหมาย เพื่อใช้ในการวน loop ในอาร์เรย์

■ ข้อสังเกตเกี่ยวกับ for each

- ❑ for each ไม่ควรนำไปใช้ในกรณี มีการเปลี่ยนแปลง collection เช่นการเพิ่มหรือลบสมาชิก
- ❑ ประสิทธิภาพของ for each อาจน้อยกว่าการวน loop แบบธรรมดา
- ❑ Collection ที่ใช้ได้กับ for each ต้องเป็น *array* หรือ Implement interface *java.lang.Iterable*

ไคอะแกรมการควบคุมการไหลของประโยค for (each)



```
for (int num : boxSize) {  
    System.out.println(num);  
}
```

การสร้าง array พร้อมกับการกำหนดค่า

- กำหนดค่าของ element ใน array เมื่อสร้างได้ เช่น

```
int[] primes = {2, 3, 5, 7, 11, 13};
```

- ซึ่งจะมีค่าเท่ากับการทำ

ไม่ต้องกำหนดขนาด

```
int[] primes;  
primes = new int[6];  
primes[0] = 2;  
:  
primes[5] = 13;
```

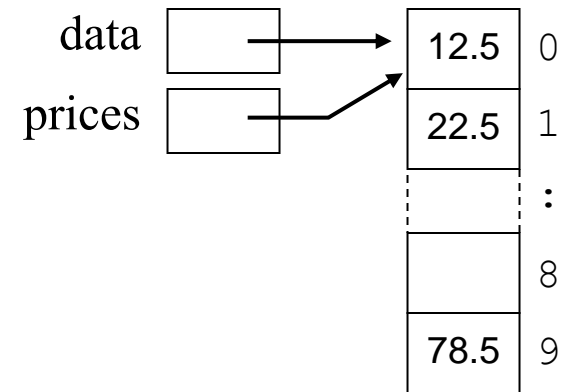
- สามารถใช้ anonymous array เพื่อส่งไปยัง method ได้โดย

```
new int[] {2, 3, 5, 7, 9, 11, 13}
```

การ copy array (1)

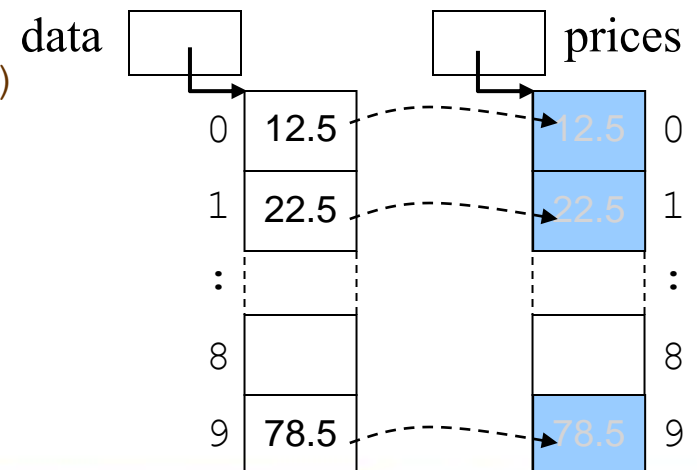
■ การสำเนา reference

```
double [] data = new double[10];  
double [] prices;  
prices = data;
```



■ การสำเนาข้อมูล

```
double [] prices =  
    new double[data.length];  
for (int i=0; i<data.length; i++)  
    prices[i] = data[i];
```



■ หรือ clone

```
double [] prices =  
    (double[])data.clone();
```

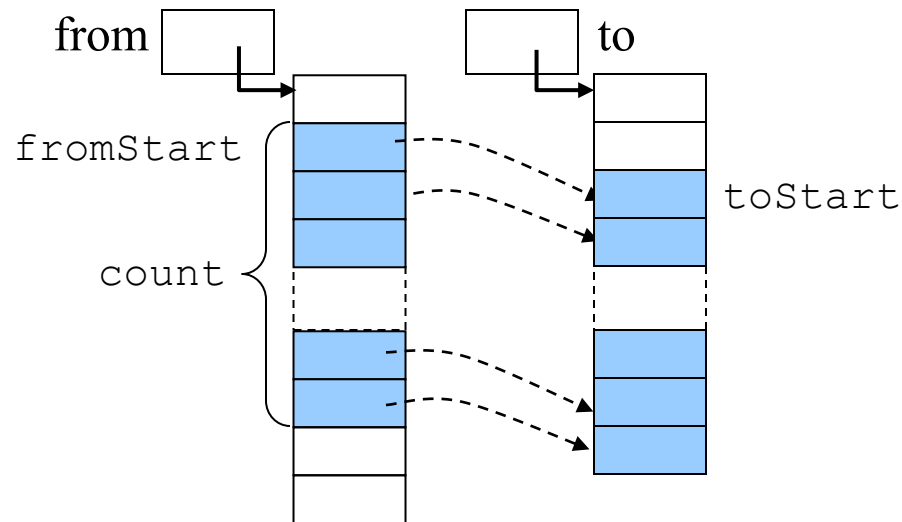
การ copy array (2)

■ การสำเนา (copy) ข้อมูลโดยใช้ **System.arraycopy**

`System.arraycopy(from, fromStart, to, toStart, count);`

□ ตัวอย่าง: ทำ copy ทั้ง array จาก data ไป prices

`System.arraycopy(data, 0, prices, 0, data.length);`



การประกาศ array โดยรู้ค่าและไม่รู้ค่าขนาดช่วงคอมไพล์

- การประกาศขนาดของ array แบบรู้ค่าในช่วง compile:

```
int[] number = new int[100];
```

- รู้ขนาดของ array ในขณะที่ compile

- การประกาศขนาดของ array แบบไม่รู้ค่าในช่วง compile:

```
int[] number;
```

```
int numberSize; ←
```

```
numberSize = scanner.nextInt();
```

```
number = new int[numberSize];
```

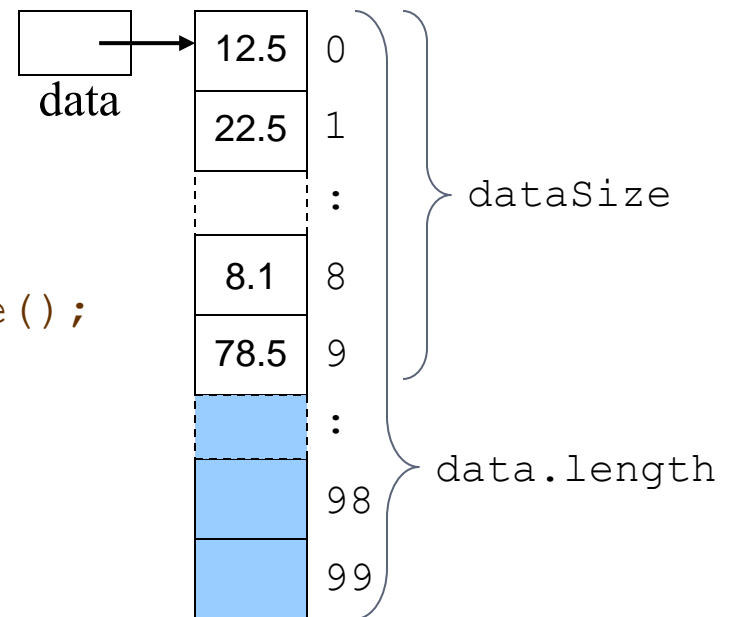
ตัวแปร **companion variable** ชื่อตัวแปร
ตามด้วยคำว่า Size เพื่อเก็บขนาด

- รู้ขนาดของ array หลังจาก run

การสร้าง array ขนาดใหญ่เกินจริงเพื่อการขยายตัว

- ในกรณีที่ไม่รู้ขนาดที่แท้จริงที่ต้องการใช้ เลือกทำได้โดย
 - ประกาศ array ขนาดใหญ่เกินจริง แล้วบรรจุค่าเท่าที่ต้องการใช้

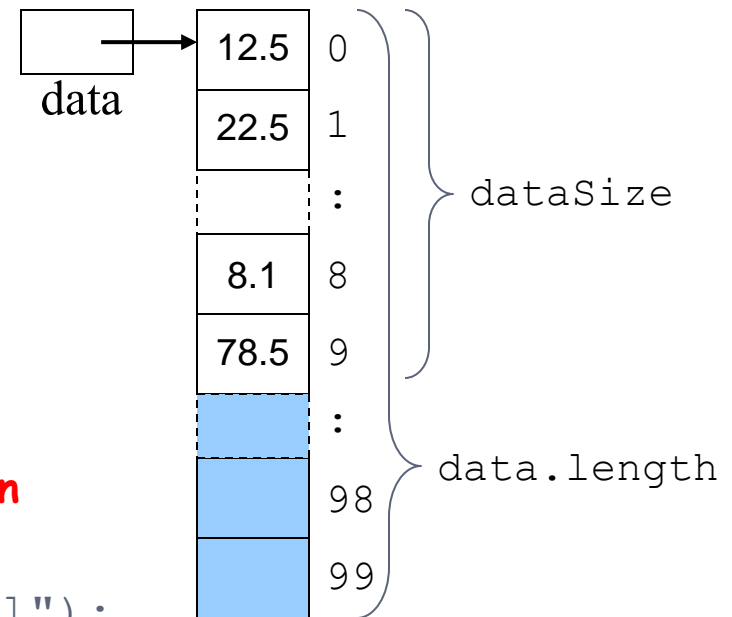
```
double [] data = new double[100];  
int dataSize = 0;  
double price = scanner.nextDouble();  
data[dataSize] = price;  
dataSize++;
```



การสร้าง array ขนาดใหญ่เกินจริงเพื่อการขยายตัว

- ในกรณีที่ไม่รู้ขนาดที่แท้จริงที่ต้องการใช้ เลือกทำได้โดย
 - ประกาศ array ขนาดใหญ่เกินจริง แล้วบรรจุค่าเท่าที่ต้องการใช้

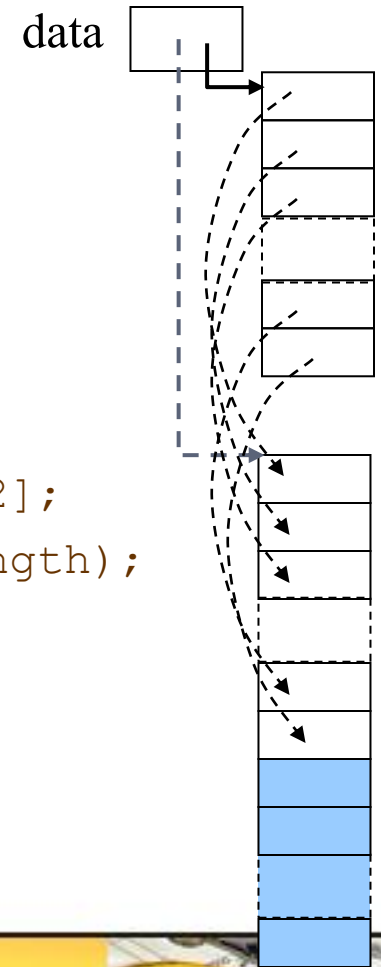
```
double [] data = new double[100];
int dataSize = 0;
double price = scanner.nextDouble();
data[dataSize] = price;
dataSize++;
// ตรวจสอบว่าขนาดเกินกว่าที่จะบรรจุได้
// เพื่อกัน error ArrayIndexOutOfBoundsException
if (dataSize >= data.length)
    System.out.println("array is full");
```



การสร้าง array ขนาดใหญ่เกินจริงเพื่อการขยายตัว (ต่อ)

- เมื่อขนาดที่กำหนดไว้ไม่เพียงพอ ขยาย array โดย
 - สร้าง array ใหม่ขนาดใหญ่ขึ้น
 - ทำ copy จาก array เก่ามายัง array ใหม่

```
if (dataSize >= data.length) {  
    // make a new array of twice the size  
    double [] newData = new double[data.length * 2];  
    System.arraycopy(data, 0, newData, 0, data.length);  
  
    // discard the old array by making  
    // data point to the newData  
    data = newData;  
}
```



ตัวอย่าง DataSet: array ในการเก็บข้อมูลเพื่อคำนวณ

- DataSet.java โดยใช้ array ในการเก็บข้อมูล เพื่อคำนวณค่าเฉลี่ย
 - ❑ `double[] data` ตัวแปรชี้ กลุ่มของตัวเลข `double`
 - ❑ `int dataSize` บอกขนาดที่ใช้ปัจจุบันของ array
 - ❑ Constructor `DataSet()`: สร้าง array ขนาด 100 ของ `double` และ set `dataSize` เป็น 0
 - ❑ `add method` กรณีพื้นที่ไม่พอ ให้ขยายขนาด array เพิ่มขึ้น 2 เท่าก่อน เพิ่มค่า `x` ลงใน array และเพิ่มค่า `dataSize` อีก 1
 - ❑ `getAverage method` คำนวณค่าเฉลี่ยของสมาชิกที่มีอยู่ใน array

ตัวอย่าง: การใช้ Array เป็น Parameters และการคืนค่า

- **average method:** คำนวณหาค่าเฉลี่ย โดยรับ array ข้อมูลเป็น parameter

```
public static double average(double [] data) {  
    if (data.length == 0) return 0;  
    double sum = 0;  
    for (int i=0; i<data.length; i++)  
        sum = sum + data[i];  
    return sum/data.length;  
}
```

อาจเขียนแบบ for each ใหม่ได้เป็น

```
for (double x : data)  
    sum = sum + x;
```

- **random method:** สร้าง array ข้อมูลที่มีค่าสุ่ม แล้วส่ง array คืนให้

```
public static int[] randomData(int length, int n) {  
    Random generator = new Random();  
    int [] data = new int[length];  
    for (int i=0; i<data.length; i++)  
        data[i] = generator.nextInt(n);  
    return data;  
}
```

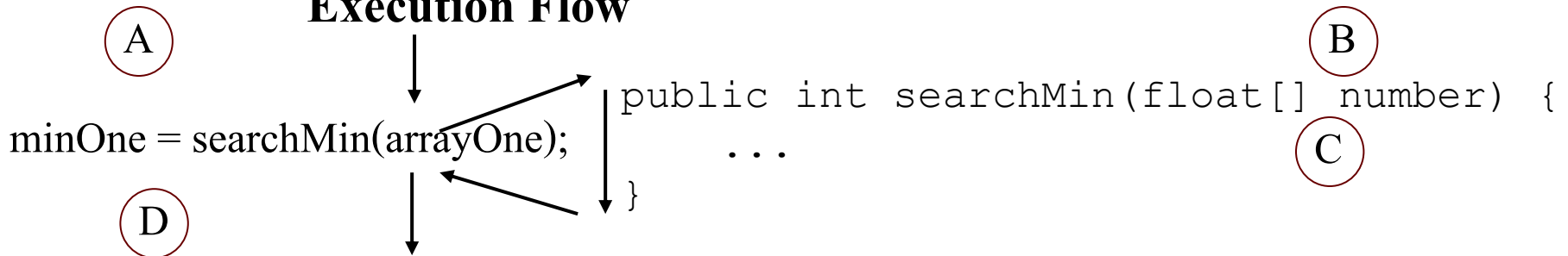
การส่ง Array เป็น Parameter ไปยัง Method

- Array เป็น object ดังนั้นการผ่าน array ไปยัง method เป็นการผ่านเพียงค่า reference ไม่มีการ copy array นั้นไปยัง method

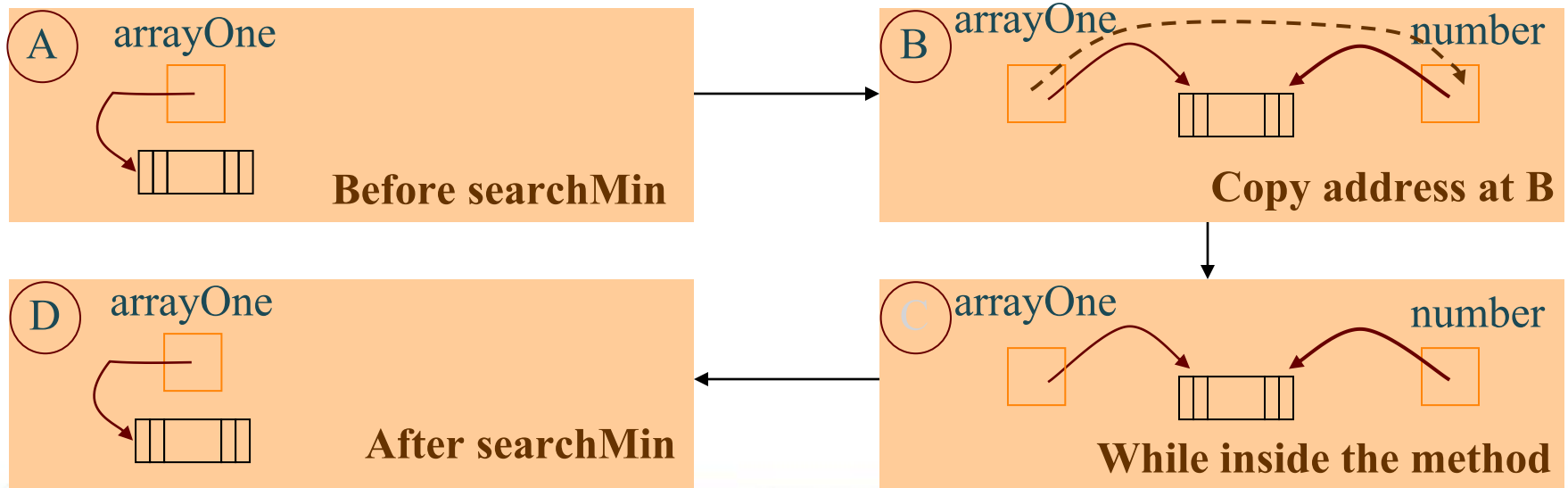
```
public int searchMin(float[] number) {  
    int indexOfMin = 0;  
    for (int j = 0; j < number.length; j++) {  
        if (number[j] < number[indexOfMin]) {  
            indexOfMin = j;  
        }  
    }  
    return indexOfMin;  
}
```

State of Memory

Execution Flow



State of Memory



ผลข้างเคียง (Side Effects)

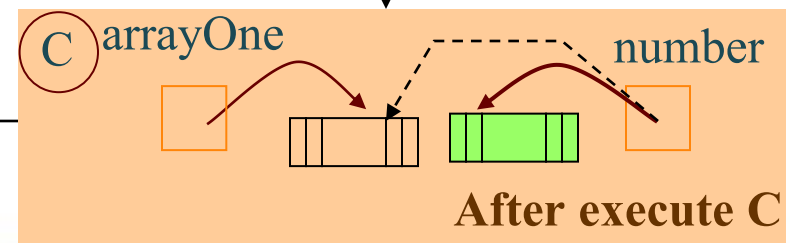
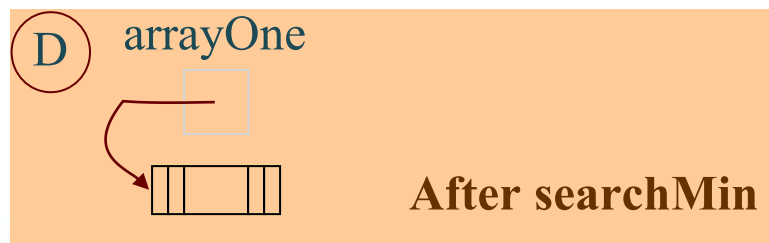
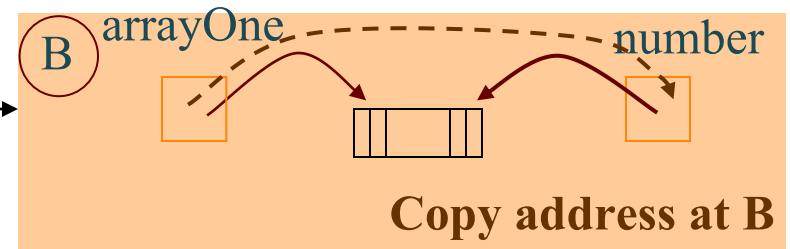
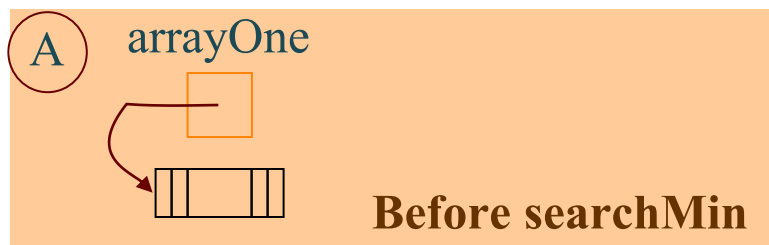
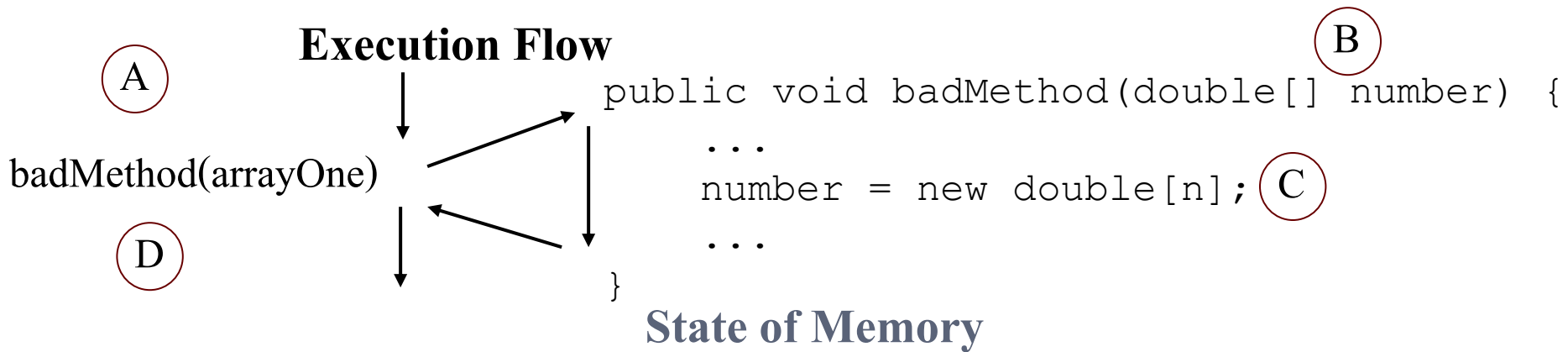
- ส่ง array ไปยัง method เพื่อให้มีผลกระทบจากการทำงานของ method

```
public void readIntegers(int[] number) {  
    for (int j = 0; j < number.length; j++) {  
        number[j] = scanner.nextInt();  
    }  
}
```

- **ระวัง!**

```
public void badMethod(double[] number) {  
    int n = scanner.nextInt();  
    number = new double[n];  
    for (int j = 0; j < number.length; j++) {  
        number[j] = scanner.nextDouble();  
    }  
}
```

สถานะของหน่วยความจำ (State of Memory)



การใช้อาร์เรย์

■ การแยกสายอักขระด้วย เมทอด split ของ String

□ คีน อาร์เรย์ของ String

```
public class Splitter {  
    public static void main(String[] args) {  
        String tester = "5342:78:40:9";  
        String [] split = tester.split(":");  
        System.out.println("Split string");  
        printArray(split);  
    }  
    private static void printArray(String[] strings){  
        for (String s:strings){  
            System.out.println("\t " + s);  
        }  
        System.out.println("-----");  
    }  
}
```


การส่งอาร์กิวเมนต์ที่มีความยาวแปรได้

- ในจาวา 5.0 ↑สามารถส่งอาร์กิวเมนต์แบบ variable-length arguments (จำนวนไม่ตายตัว) ได้

- Syntax:

```
returnType methodName(type... variableName) { ... }
```

เมื่อ

- ☐ type เป็นชนิดข้อมูล/คลาส
- ☐ เครื่องหมาย ... หลังชื่อชนิดเป็นเครื่องหมายบ่งบอกว่าเป็น variable-length
- สามารถมีตัวแปรมากกว่า 1 ตัว โดยที่ variable-length arguments ต้องอยู่เป็นตัวสุดท้ายของรายการพารามิเตอร์เสมอ

ตัวอย่าง

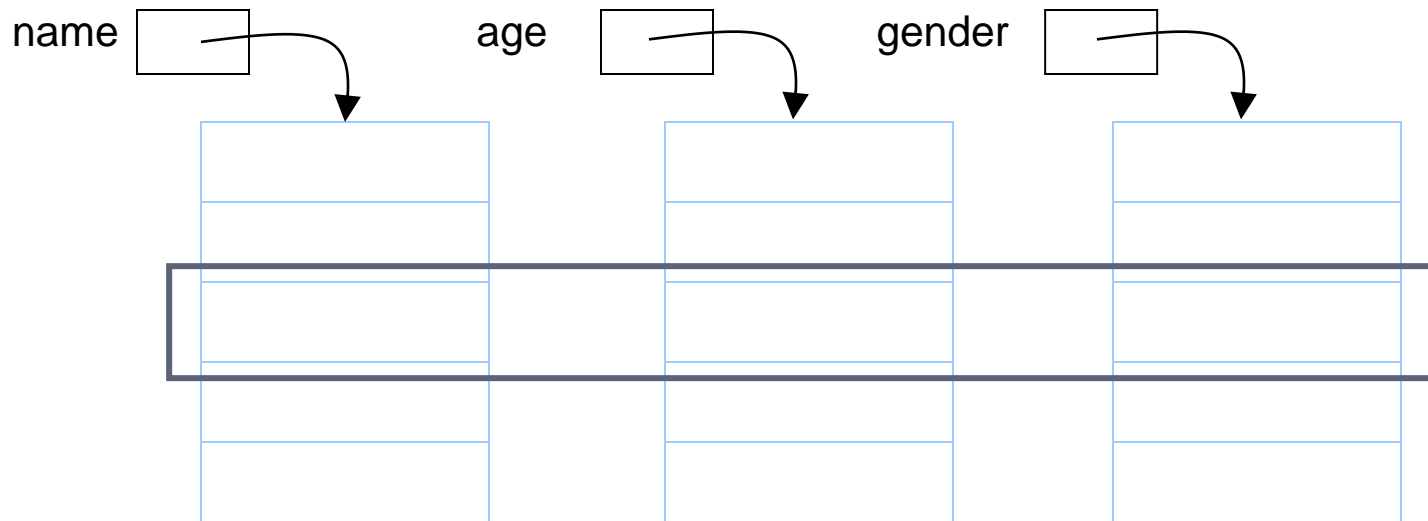
```
public class TestVarArgs {
    public static double average(int ... vararg) {
        double total = 0.0;
        for (int i: vararg)
            total += i;

        return vararg.length > 0? total/vararg.length : 0.0;
    }

    public static void main(String... args) {
        System.out.println("Average " + average(2, 3, 4));
        System.out.println("Average " + average(3, 4));
        System.out.println("Average " + average());
    }
}
```

Array ของข้อมูลที่มีความสัมพันธ์กัน

- ตัวอย่าง: ข้อมูลของบุคคล มีชื่อ อายุ และเพศ
- อาจเลือกสร้าง parallel array เพื่อเก็บค่าทั้ง 3 อย่างไว้และพยายามรักษา index ให้ตรงกันเสมอ → สร้างปัญหายุ่งยากในการบำรุงรักษา!!



ทางแก้

- สร้าง Object ของข้อมูล และเก็บ Objects ใน array เมื่อต้องการใช้ข้อมูลเป็นกลุ่ม

```
class Person {  
    private String name;  
    private int age;  
    private char gender;  
    public Person() {  
        name = "Not Given"; age = 0; gender='U';  
    }  
}
```

- ถาม ข้อดีของการทำลักษณะนี้?

- พิจารณา กรณีเพิ่มข้อมูลเงินเดือนของบุคคล

Person Object

Design Document: Person Class

method	Purpose
public int getAge()	คืนค่าอายุของคนนั้น ค่า default ถูก set เป็น 0
public char getGender()	คืนค่าเพศของคนนั้น โดยค่าที่เป็นไปได้คือ F, M, U
public String getName()	คืนค่าชื่อของคนนั้น default name คือ Not Given
public void setAge(int age)	Set ค่าอายุของคนนั้น
public void setGender(char g)	Set ค่าเพศ
public void setName(String n)	Set ค่าของชื่อ

Array of Objects

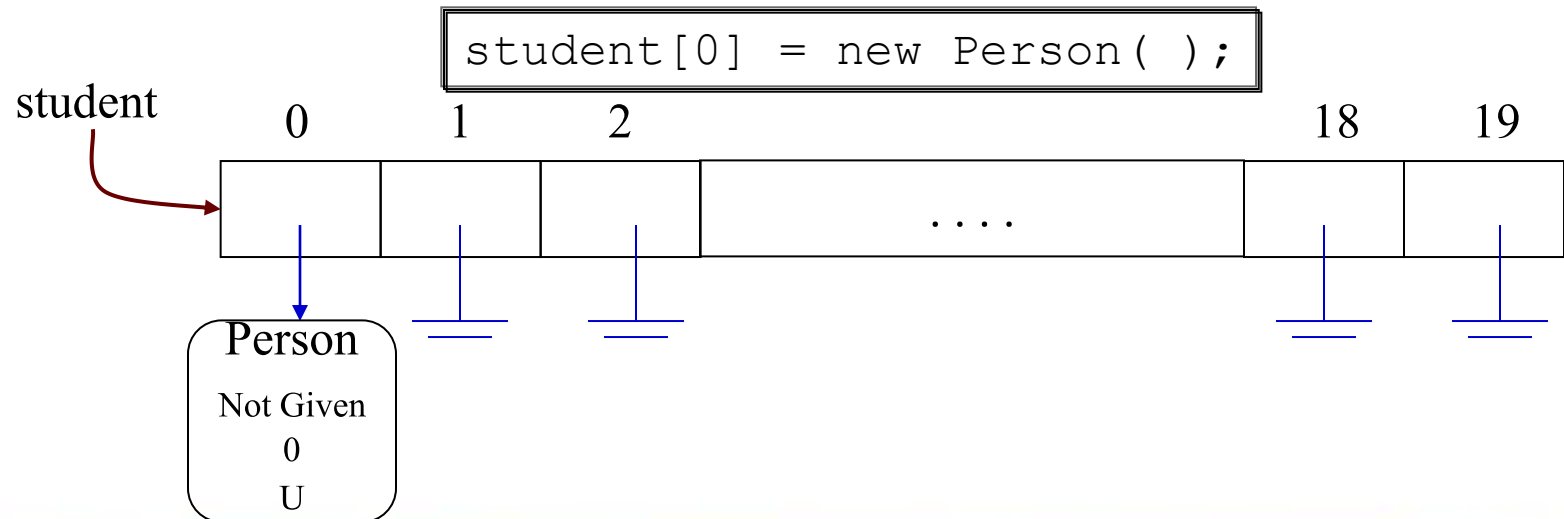
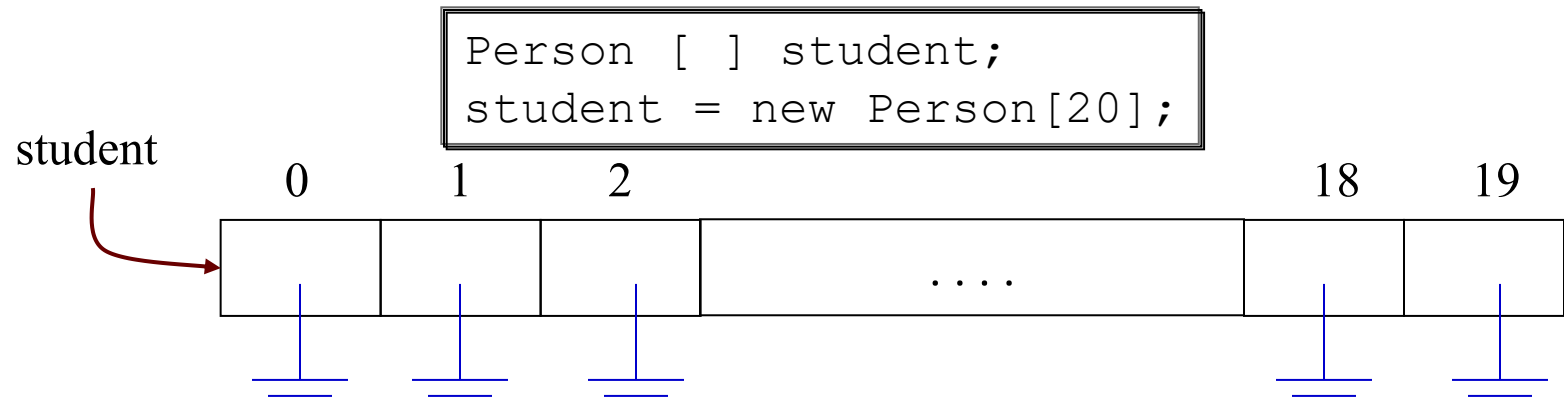
■ สร้าง nidoi Person object 1 คน

```
Person nidoi;  
nidoi = new Person( );  
nidoi.setName("Noo Nid Noi");  
nidoi.setAge(3);  
nidoi.setGender('F');  
System.out.println("    Name: " +nidoi.getName());  
System.out.println("    Age: " +nidoi.getAge());  
System.out.println("Gender: " +nidoi.getGender());
```

■ หากต้องการหลาย ๆ คนเพื่อใช้กับนักศึกษา:

```
Person [ ] student;           //declare the person array  
student = new Person[20];     //create it
```

Array of Person



Multidimensional Array

- Multidimensional array สามารถสร้างได้จากอาร์เรย์ของอาร์เรย์
- การประกาศ array 2 มิติ (2-dimensional array):

- ❑ `float [] [] payScaleTable; หรือ`

- ❑ `float payScaleTable [] [];`

- สร้าง array ได้โดย

- ❑ `payScaleTable = new float [4] [5];`

`payScaleTable[2][1]`

	0	1	2	3	4
0					
1					
2					
3					

payScaleTable = new float [4][5]

■ เป็นการรวมการทำงานของขั้นตอนต่อไปนี้

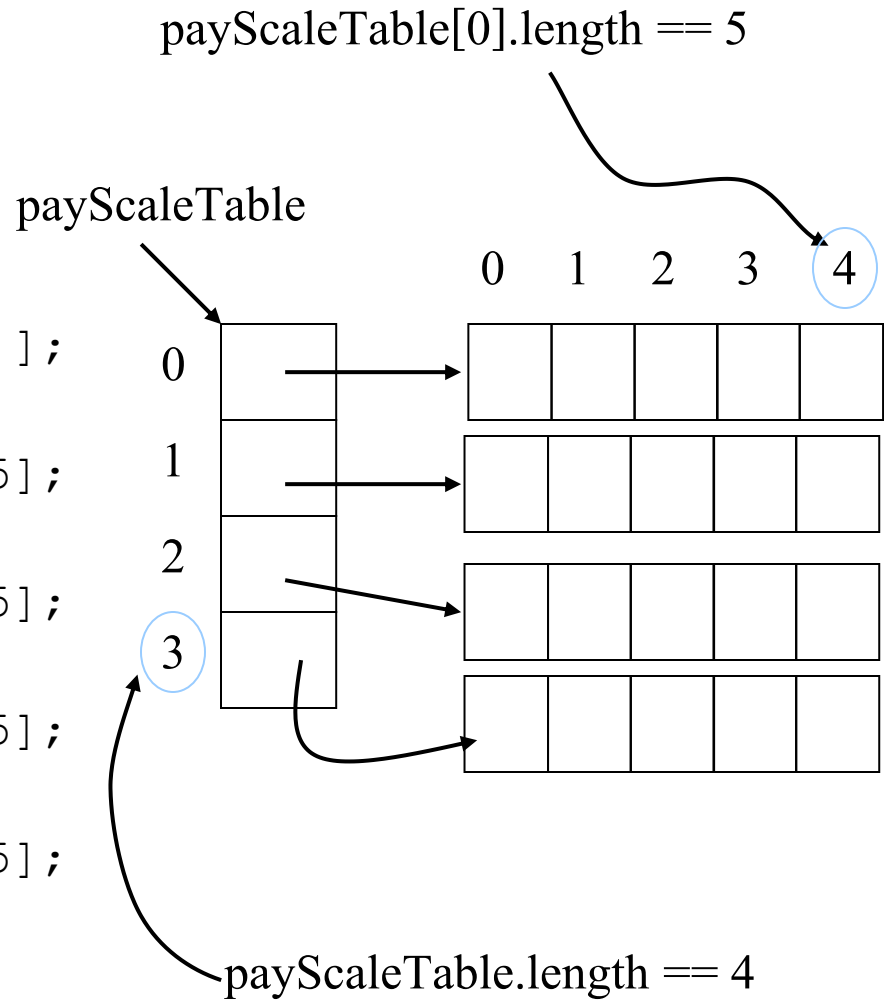
```
payScaleTable = new float[4][ ];  
payScaleTable[0] = new float[5];  
payScaleTable[1] = new float[5];  
payScaleTable[2] = new float[5];  
payScaleTable[3] = new float[5];
```

■ หรือ

```
payScaleTable = new float[4][ ];  
for (int j = 0; j < 4; j++)  
    payScaleTable[j] = new float[5];
```

การทำงานที่เกิดขึ้น

```
payScaleTable = new float[4][ ];  
payScaleTable[0] = new float[5];  
payScaleTable[1] = new float[5];  
payScaleTable[2] = new float[5];  
payScaleTable[3] = new float[5];
```



subarray

- Subarray = array ซึ่งเป็นส่วนหนึ่งของอีก array
- payScaleTable มี 4 subarrays ที่มีขนาดเท่ากัน
- อาจสร้าง subarray ขนาดที่ต่างกันก็ได้ เช่น

```
mixArray = new float[4] [ ];  
mixArray[0] = new float[3];  
mixArray[1] = new float[6];  
mixArray[2] = new float[4];  
mixArray[3] = new float[2];
```

	0	1	2	3	4	5
0						
1						
2						
3						

Collection: ArrayList

- **ลิสต์ (List):** เก็บสมาชิกแบบมีลำดับ (*Sequence*) และมีขนาดจำกัด (finite) โดยอาจมีสมาชิกซ้ำกันได้
- ArrayList เป็น Collection ที่ใช้เก็บสมาชิกแบบ list มีข้อต่างจากอาร์เรย์
 - สามารถเพิ่มและลดขนาดได้อัตโนมัติ
 - มีเมทอดช่วยในการจัดการกับสมาชิก เช่นการเพิ่ม ลบสมาชิก ได้

Array List (1)

- ArrayList เป็น class ใน `java.util` package

- ArrayList ใช้เพื่อเก็บค่ากลุ่มลำดับของวัตถุชนิด E

- การประกาศตัวแปร: `ArrayList<E> variableName;`

- การสร้าง: `new ArrayList<E>();`

- ตัวอย่างการประกาศและสร้าง

```
ArrayList<String> stringList = new ArrayList<String>();
```

- สังเกต เมื่อสร้างต้องระบุชนิดของสมาชิกที่จะเก็บใน list โดยใส่ชื่อชนิดระหว่าง `<>`

เมทอดของ ArrayList<E> (บางส่วน)

- `boolean add(E e)`: เพิ่ม element เข้าท้าย list (optional operation)
- `void add(int index, E e)`: เพิ่ม element ที่ตำแหน่งที่ระบุ
- `boolean addAll(Collection<? extends E> c)`: เพิ่มทุก elements ใน collection เข้าท้าย list ตามลำดับที่ถูกคืนจาก iterator ของ collection นั้น
- `void clear()`: ลบทุก elements ออกจาก list
- `boolean contains(Object o)`: คืนจริงถ้าพบ element นั้นใน list
- `E get(int index)`: คืน element ที่อยู่ใน list ตามตำแหน่งที่ระบุ
- `int indexOf(Object o)`: คืนค่า index ที่พบตัวแรกของ element นั้นใน list, หรือ -1 ถ้าไม่พบ
- `boolean isEmpty()`: คืนค่าจริงถ้า list ไม่มี elements ใด ๆ
- `int lastIndexOf(Object o)`: คืนค่า index ตัวสุดท้ายของ element นั้นใน list หรือ -1 ถ้าไม่พบ
- `E remove(int index)`: ลบ element ในตำแหน่งที่ระบุ
- `E set(int index, E e)`: เปลี่ยน element ตำแหน่งที่ระบุด้วย element ใหม่ที่ให้
- `int size()`: คืนค่าจำนวน elements ที่มีใน list

ตัวอย่างเมทอดขนาดและการเข้าถึงสมาชิก

- **การหาขนาด:** size method บอกจำนวนสมาชิกใน ArrayList

```
stringList.size();
```

- **การดึงสมาชิกออกจาก ArrayList:** get method

- แต่ละวัตถุใน array list จะมีตำแหน่งลำดับเป็น int เรียกว่า index

- ตำแหน่ง (index) ของสมาชิกอยู่ระหว่าง 0 ถึง size() - 1

- ใช้ get method พร้อมระบุตำแหน่ง เพื่อดึงสมาชิกที่ต้องการ

```
String name = stringList.get(0);
```

- ข้อผิดพลาดที่พบบ่อย การอ่านเกินตำแหน่ง

```
int n = stringList.size();
```

```
String name = stringList.get(n);
```

ตัวอย่างเมทอดสำหรับการปรับปรุงสมาชิก

■ การปรับปรุง

- เปลี่ยนสมาชิกโดยระบุตำแหน่ง

```
stringList.set(0, "Zombie");
```

- เพิ่มสมาชิก โดย add method

- เพิ่มต่อท้าย

```
stringList.add("Monster");
```

- เพิ่มแทรกแบบระบุตำแหน่ง

```
stringList.add(0, "Dragon");
```

- ลบสมาชิก

```
stringList.remove(0);
```


การวน loop เพื่อเข้าถึงสมาชิกใน Array List

- ArrayList สามารถวน loop ในแบบ **for each** ได้เช่นกัน

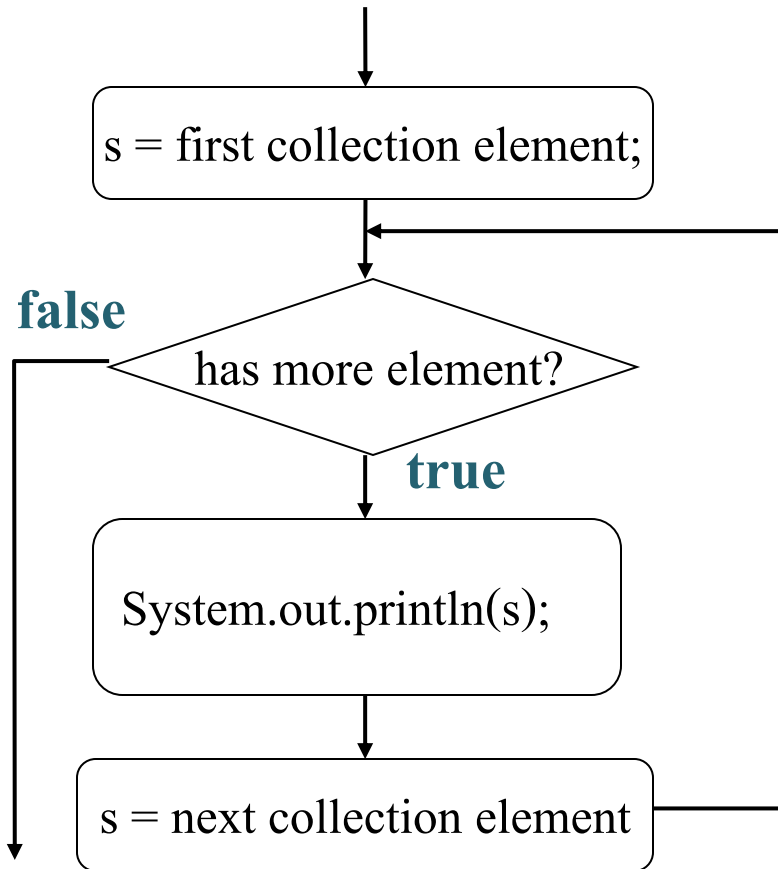
```
for (FormalParameter : Expression)  
    Statements;
```

- ตัวอย่างเช่น

```
for (String s: stringList) { ... }
```

- จุดมุ่งหมายเพื่อใช้ในการวน loop ใน Collection

ไคอะแกรมการควบคุมการไหลของประโยค for (each)



```
for (String s: stringList) {  
    System.out.println(s);  
}
```

ตัวอย่างการอ่านค่าแต่ละสมาชิกใน Array List

■ อ่านค่าสมาชิกแต่ละตัว โดยการวน loop ระบุ index

```
for (int i=0; i < stringList.size(); i++) {  
    String s = (String) stringList.get(i);  
    ... // do something with s  
}
```

■ อ่านค่าสมาชิกแต่ละตัว โดยการวนแบบ for each

```
for (String s: stringList) {  
    ... // do something with s  
}
```

ตัวอย่าง MonsterWorld.java: การใช้ ArrayList กับ MonsterWorld

- class MonsterWorld เก็บกลุ่มของ Monster
 - ❑ ArrayList<Monster> world เป็น attribute เพื่อเก็บ collection ของ monster
 - ❑ Constructor MonsterWorld() สร้าง ArrayList
 - ❑ เมทอด add เพิ่มวัตถุ monster ใน world
 - ❑ เมทอด remove ลบวัตถุ monster ณ ตำแหน่งที่ระบุ
 - ❑ เมทอด print พิมพ์สัตว์ประหลาดทั้งหมดที่อยู่ใน MonsterWorld
 - ❑ เมทอด count คำนวณจำนวน monster ทั้งหมดใน world
 - ❑ เมทอด getCoverage คำนวณพื้นที่ทั้งหมดที่ monster เห็นรวมทั้งหมด
 - ❑ เมทอด getAverage คำนวณหาพื้นที่เฉลี่ยของการมองเห็นของ monster

```
import java.util.ArrayList;

/** A MonsterWorld holds a collection of monster. */
public class MonsterWorld {
    private ArrayList<Monster> world;

    /** Constructs an empty world. */
    public MonsterWorld() {
        world = new ArrayList<>();
    }

    /** Add a monster to the world.
     * @param mons the monster to add
     */
    public void add(Monster mons) {
        world.add(mons);
    }

    /** Counts the number of monsters in the world
     * @return the number of monsters
     */
    public int count() {
        return world.size();
    }
}
```

```
/** Compute the total coverage area that monsters see.  
    @return the sum of coverage area  
*/  
public double getCoverage(){  
    double sum = 0;  
    for (Monster m: world){  
        sum += m.getArea();  
    }  
    return sum;  
}  
  
/** Compute average area that monsters see.  
    @return the average value of area  
*/  
public double getAverage(){  
    if(world.size() == 0) return 0;  
    return getCoverage()/world.size();  
}
```

Primitive Wrapper Class

- ArrayList ไม่สามารถใช้เก็บชนิดพื้นฐานได้
- **Primitive Wrapper Class**
 - 8 ชนิดพื้นฐานในจาวา มี 8 คลาสที่สอดคล้องกันอยู่ในแพ็คเกจ java.lang
 - ชื่อสอดคล้องกัน แต่ขึ้นต้นด้วยตัวพิมพ์ใหญ่ (ยกเว้นชื่อของ int กับ char)
 - double → Double, int → Integer, char → Character
 - ทั้งหมดเป็น immutable class
- ในจาวา 5.0↑ ทำ implicit conversion ระหว่าง wrapper objects และค่าของชนิดพื้นฐานเหล่านี้ (ทั้งแปลงไปและกลับ) เรียกวิธีการแปลงนี้ว่า

Autoboxing

Primitive Wrapper Class

Primitive Type	Wrapper class
int	Integer
byte	Byte
short	Short
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

```
int number = 5;  
float number = 2.0F;  
boolean check = true;  
-----
```

```
Integer number = new Integer(10);  
Float number = new Float(2.0F);  
Boolean check = new Boolean(true);  
Boolean check = Boolean.TRUE;
```


Primitive Wrapper Class

```
int number = 5;
```

```
Integer x = new Integer(number); //Boxing
```

```
Integer x = number; //Autoboxing
```

```
Integer x = new Integer(5);
```

```
int number = x.intValue(); //Unboxing
```

```
int number = x; //Autounboxing
```

```
import java.util.Random;
import java.util.ArrayList;

public class WrapperList {
    private ArrayList<Double> dList;

    public WrapperList() {
        dList = new ArrayList<Double>();
    }

    public void add(double d) {
        dList.add(d);
    }

    public void printList() {
        for (int i=0; i<dList.size(); i++)
            System.out.println(dList.get(i));
    }

    public double getTotal() {
        double total = 0;
        for (Double d: dList)
            total += d;
        return total;
    }
}
```

สรุปการเรียนรู้ในวันนี้

- เข้าใจแนวคิดเกี่ยวกับ Collection
- รู้จักการใช้ array และ array list เพื่อการเก็บกลุ่มของข้อมูล
- สามารถประกาศคลาสที่มี array และ array list เป็นสมาชิก
- อธิบายการสร้าง array 2 มิติ (2-dimensional) จากอาร์เรย์ของอาร์เรย์
- เรียนรู้เกี่ยวกับ algorithm พื้นฐานเพื่อทำงานกับกลุ่มของข้อมูล เช่นการค้นห การจัดการกับสมาชิกใน Collection
 - เข้าถึงหน่วยย่อยในกลุ่มข้อมูลที่ถูกเก็บใน array, array lists
 - implement array เพื่อการขยาย โดยเก็บค่าเพียงบางส่วนก่อน
 - การส่งผ่าน array ไปยัง methods