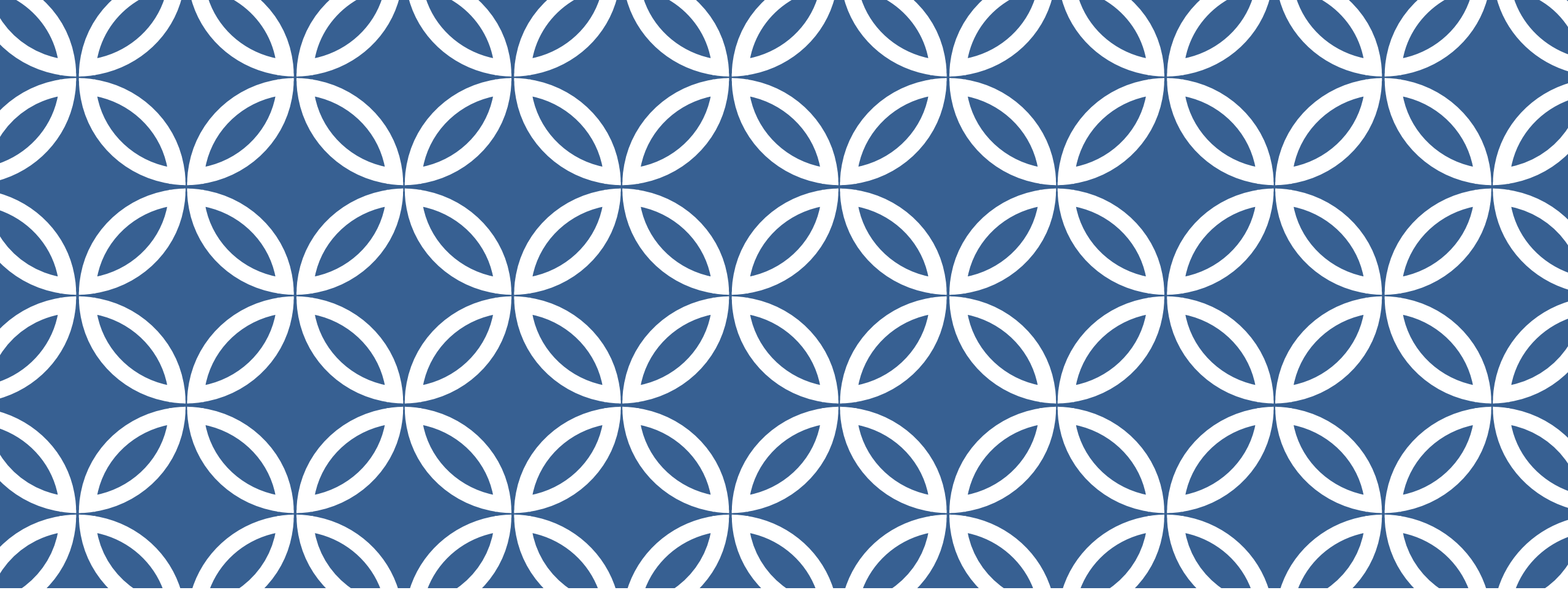


บทที่ 2
IMPLEMENTATION IN JAVA

อ.สกรณ์ บุษบง
สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
มหาวิทยาลัยราชภัฏบุรีรัมย์

METHOD IMPLEMENTATION

- การเขียนโปรแกรมโดยปกติจะเป็นการสร้าง **method** เพื่อใช้งาน **method** ของ **java** นั้นประกอบไปด้วย
- access returntype name (parameter,...){}

METHOD IMPLEMENTATION > ACCESS

access คือการกำหนดการเข้าถึงของ **method** ซึ่งประกอบไปด้วย 3 รูปแบบหลักๆ

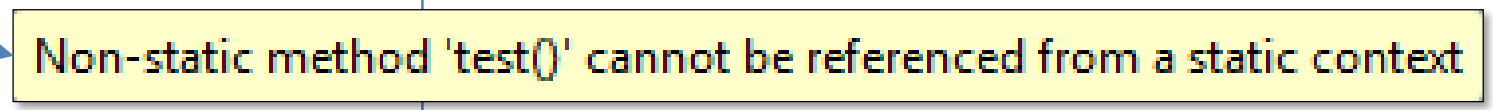
- public
- protected
- private

	Class	Package	Subclass same pkg	Subclass diff pkg	world
public	+	+	+	+	+
protected	+	+	+	+	O
private	+	O	O	O	O

METHOD IMPLEMENTATION > ACCESS

Static method???

```
package com.company;

public class Main {

    public static void main(String[] args) {
        test();
    }
    public void test(){

    }
}
```



STATIC METHOD

- การเขียนโปรแกรมเชิงวัตถุนั้น **method** ใน **class** จะทำงานต่อเมื่อ **object** ของ **class** เรียกใช้งาน
- การใช้ **static** กับ **method** ทำให้ **method** นั้นๆสามารถทำงานได้โดยไม่ต้องสร้าง **object**
- ยกตัวอย่างเช่น

```
public static void main(String[] args)
```

METHOD IMPLEMENTATION > ACCESS > PUBLIC

```
package com.company;

public class Main {

    public static void main(String[] args) {
        test();
    }
    public static void test(){

    }
}
```

method main เป็น static method ดังนั้น method ใดๆที่ถูก main เรียกใช้โดยตรงต้องเป็น static ด้วย

METHOD IMPLEMENTATION > ACCESS > PUBLIC

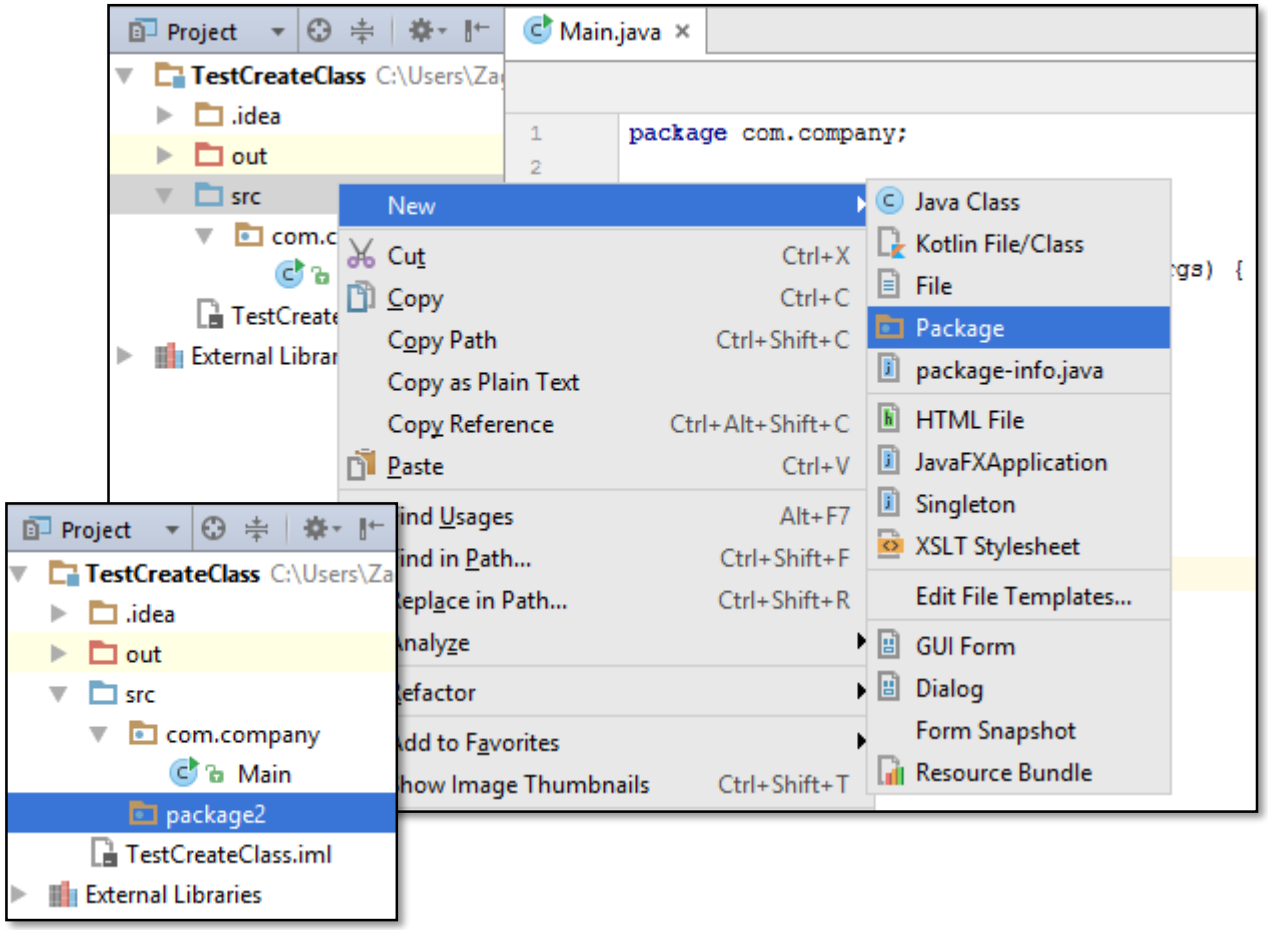
```
package com.company;

public class Main {

    public static void main(String[] args) {
        Main main = new Main();
        main.test();
    }
    public void test(){

    }
}
```

ถ้าอยากเรียกใช้ **method** โดยไม่กำหนดให้เป็น **static** ก็สามารถทำได้โดย **new object** ของ **class main** ก่อน จากนั้นให้ **object** เป็นผู้เรียกใช้ **method**

METHOD IMPLEMENTATION > ACCESS > PROTECTED

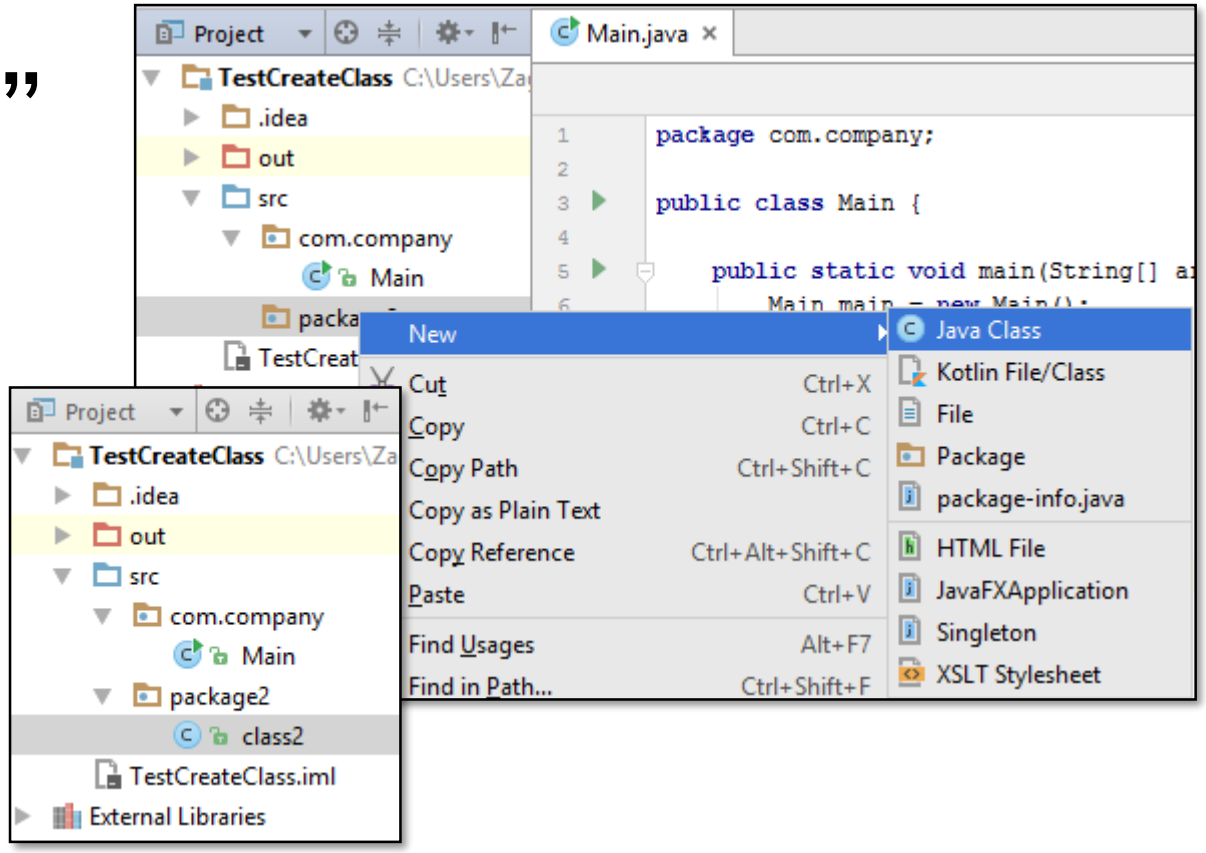
สร้าง package ใหม่

ใช้ชื่อว่า “package2”



METHOD IMPLEMENTATION > ACCESS > PROTECTED

สร้าง class ใหม่ ชื่อ “class2”



METHOD IMPLEMENTATION > ACCESS > PROTECTED

```java
package com.company;
import package2.class2;
public class Main {

    public static void main(String[] args)
{
        /*Main main = new Main();
        main.test();*/
        class2 c2 = new class2();
        c2.method1();
    }
    public void test(){

    }
}
```

```java
package package2;

public class class2 {
    protected void method1(){

    }
}
```

'method1()' has protected access in 'package2.class2'

METHOD IMPLEMENTATION > ACCESS > PROTECTED

```java
package com.company;
import package2.class2;
public class Main {

    public static void main(String[] args)
{
        /*Main main = new Main();
        main.test();*/
        class2 c2 = new class2();
        c2.method1();
    }
    public void test(){

    }
}
```

```java
package package2;

public class class2 {
    public void method1(){

    }
}
```

METHOD IMPLEMENTATION > ACCESS > PRIVATE

```java
package com.company;
import package2.class2;
public class Main {

    public static void main(String[] args)
{
        Main main = new Main();
        main.test();
        class2 c2 = new class2();
        c2.method1();
    }
    private void test(){

    }
}
```

```java
package package2;

public class class2 {
    private void method1(){

    }
}
```

'method1()' has private access in 'package2.class2'

METHOD IMPLEMENTATION > ACCESS

- ทั้ง **public, protected, private** นั้นบ่งบอกถึงขอบเขตการเข้าถึงข้อมูล ซึ่งสามารถใช้ได้กับ **class, method** และ **instance** ได้
- ลักษณะแบบนี้เป็นคุณสมบัติของ OOP เรียกว่า

  Information Hiding
  - ทำให้เราสามารถจำกัดการเข้าถึง
  - **Class** อื่นๆ ไม่สามารถเข้าถึง **method** ของอีก **class** ได้

METHOD IMPLEMENTATION > RETURNTYPE

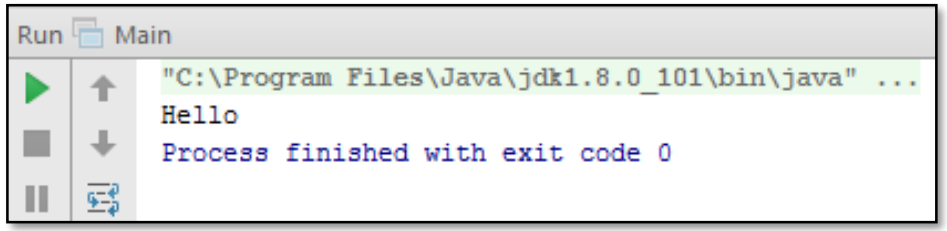
access returntype name (parameter,…){}

- returntype คือสั่งที่บ่งบอกว่า เมื่อ **method** ทำงานเสร็จแล้วจะได้ผลลัพธ์อะไรกลับไป
- returntype สามารถเป็น **type** พื้นฐานได้ หรือแม้กระทั้ง เป็น **object** ก็ได้

METHOD IMPLEMENTATION > RETURNTYPE

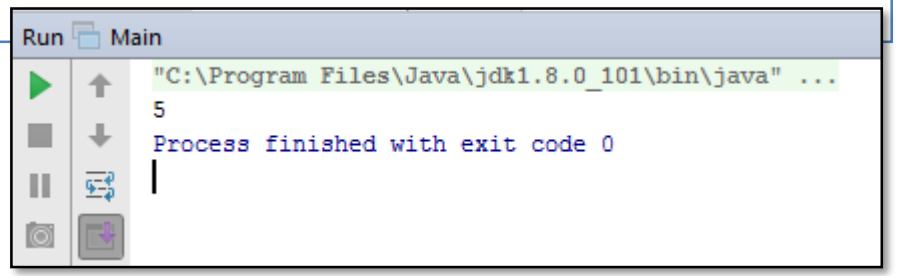
```
package com.company;

public class Main {

    public static void main(String[] args) {
        Main main = new Main();
        main.test();
    }
    public void test(){
        System.out.print("Hello");
    }
}
```

- Method test() ได้กำหนด returntype เป็น void
- void คือการกำหนดให้ method นั้นๆไม่ต้องคืนค่ากลับ



METHOD IMPLEMENTATION > RETURNTYPE

```
package com.company;

public class Main {

    public static void main(String[] args) {
        Main main = new Main();
        //main.test();
        System.out.print(main.test2());
    }
    public void test(){
        System.out.print("Hello");
    }
    public int test2(){
        return  5;
    }
}
```



- Method test() ได้กำหนด returntype เป็น int
- ดังนั้น method test() จะต้องคืนค่าที่เป็น type int เท่านั้น

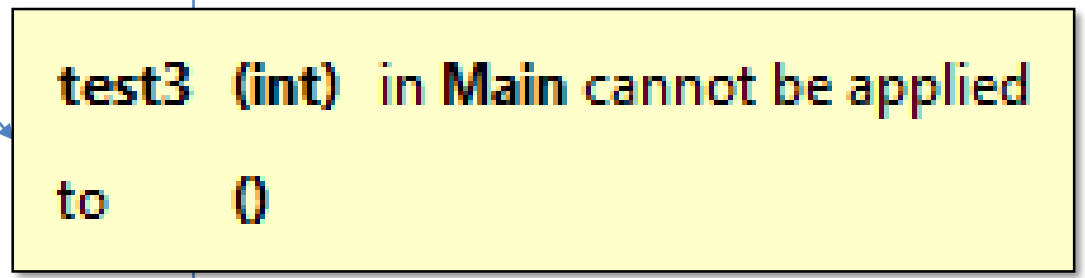
METHOD IMPLEMENTATION > PARAMETER

- access return_type name (parameter,...){}
  - parameter คือค่าที่จำเป็นในการทำงานของ **method** นั้นๆ
  - parameter มีได้ตั้งแต่ 0 ถึง 255(**java 4.4.3**)
  - เมื่อ **method** ถูกใช้งานจะต้องส่งค่า parameter ให้ครบตามจำนวน
  - parameter นั้นจะเป็นตัวแปรพื้นฐาน หรือ **object** ก็ได้

METHOD IMPLEMENTATION > PARAMETER

```
package com.company;

public class Main {

    public static void main(String[] args) {
        Main main = new Main();
        //main.test();
        //System.out.print(main.test2());
        int a = main.test3();
        System.out.print(a);
    }
    public void test(){
        System.out.print("Hello");
    }
    public int test2(){
        return  5;
    }
    public int test3(int a){
        return a;
    }
}
```

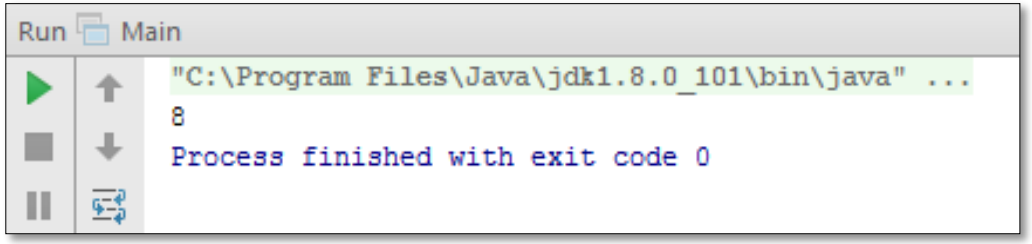
ถ้าไม่ใส่ parameter ไม่ถูกต้องก็จะไม่สามารถทำงานได้



METHOD IMPLEMENTATION > PARAMETER

```
package com.company;

public class Main {

    public static void main(String[] args) {
        Main main = new Main();
        //main.test();
        //System.out.print(main.test2());
        //int a = main.test3(6);
        //System.out.print(a);
        System.out.print(main.test4(6,3));
    }
    public void test(){
        System.out.print("Hello");
    }
    public int test2(){
        return  5;
    }
    public int test3(int a){
        return a;
    }
    public int test4(int a, int b ){
        return a+b;
    }
}
```

- การกำหนดจำนวน parameter มีกี่ตัวก็ได้ ตั้งแต่ 0 - 255



METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

การผสมผสานระหว่าง returntype และ parameter ทำให้เกิดรูปแบบการเข้าถึง **attribute** ที่เราสามารถกำหนดได้ว่าจะให้เป็น

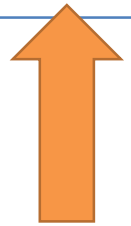
- Read-Only
- Write-Only
- Read-Write

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

สร้าง class3 ใน package2

```
package package2;

public class class2 {
    public void test_class2(){
        class3 c3 = new class3();
        c3.a = 4;
    }
}
```

```
package package2;

public class class3 {
    int a =5;
}
```



package-private, read/writable for all classes inside same package

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

```
package com.company;

import package2.class3;

public class Main {

    public static void main(String[] args)
{
        Main main = new Main();
        class3 c3 = new class3();
        c3.a = 4;
    }
…
…
}
```

```
package package2;

public class class3{
    int a =5;
}
```

'a' is not public in 'package2.class3'. Cannot be accessed from outside package

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

```
package com.company;

import package2.class3;

public class Main {

    public static void main(String[] args)
{
        Main main = new Main();
        class3 c3 = new class3();
        c3.a = 4;
    }
…
…
}
```

```
package package2;

public class class3 {
    public int a =5;
}
```

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

- การเขียนโปรแกรมแบบ OO มักจะไม่กำหนดให้ **attribute** เป็น **public** โดยไม่จำเป็น เพราะเสี่ยงต่อการเข้าถึงจาก Object อื่น
- ดังนั้นเราควรกำหนด **attribute** ให้เป็น **protected** หรือ **private** แทน แล้วจำกัดการเข้าถึงผ่าน **method**

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

ดังนั้น class3 ควรเป็นแบบนี้

```
package com.company;

import package2.class3;

public class Main {

    public static void main(String[] args)
{
        Main main = new Main();
        class3 c3 = new class3();
        c3.a = 4;
    }
…
…
}
```

```
package package2;

public class class3 {
    private int a =5;
}
```

'a' has private access in 'package2.class3'

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

- แล้ว **class main** จะเข้าถึงตัวแปร **a** ใน **class3** ได้ยังไง
- สมมุติว่าเรากำหนดให้ตัวแปร **a** ใน **class3** เป็น Read-Only Attribute

```
package package2;

public class class3 {
    private int a =5;
    public int geta(){
        return a;
    }
}
```

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

การเข้าถึง Read-Only Attribute ทำได้ผ่าน method geta ของ class3

```
package com.company;

import package2.class3;

public class Main {

    public static void main(String[] args) {
        Main main = new Main();
        class3 c3 = new class3();
        System.out.print(c3.geta());
    }
...
}
```

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

ถ้าอยากกำหนดค่าให้ตัวแปร a ใน class3 ให้มีคุณสมบัติ Read-Write Attribute ต้องเพิ่ม methodดังนี้

```
package package2;

public class class3 {
    private int a =5;
    public int geta(){
        return a;
    }
    public void seta(int a){
        this.a = a;
    }
}
```

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

การเรียกใช้จะมีลักษณะดังนี้

```
package com.company;

import package2.class3;

public class Main {

    public static void main(String[] args) {
        Main main = new Main();
        class3 c3 = new class3();
        c3.seta(10);
        System.out.print(c3.geta());
    }
...
}
```

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

ถ้าอยากให้ตัวแปร a ใน class3 เป็น Write-Only Attribute ทำยังไง ?????

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

การกำหนด

- Read-Only Attribute
- Write-Only Attribute
- Read-Write Attribute

เป็นเรื่องของการใช้ Getter และ Setter

METHOD IMPLEMENTATION > COMBINATION OF RETURNTYPE AND PARAMETER

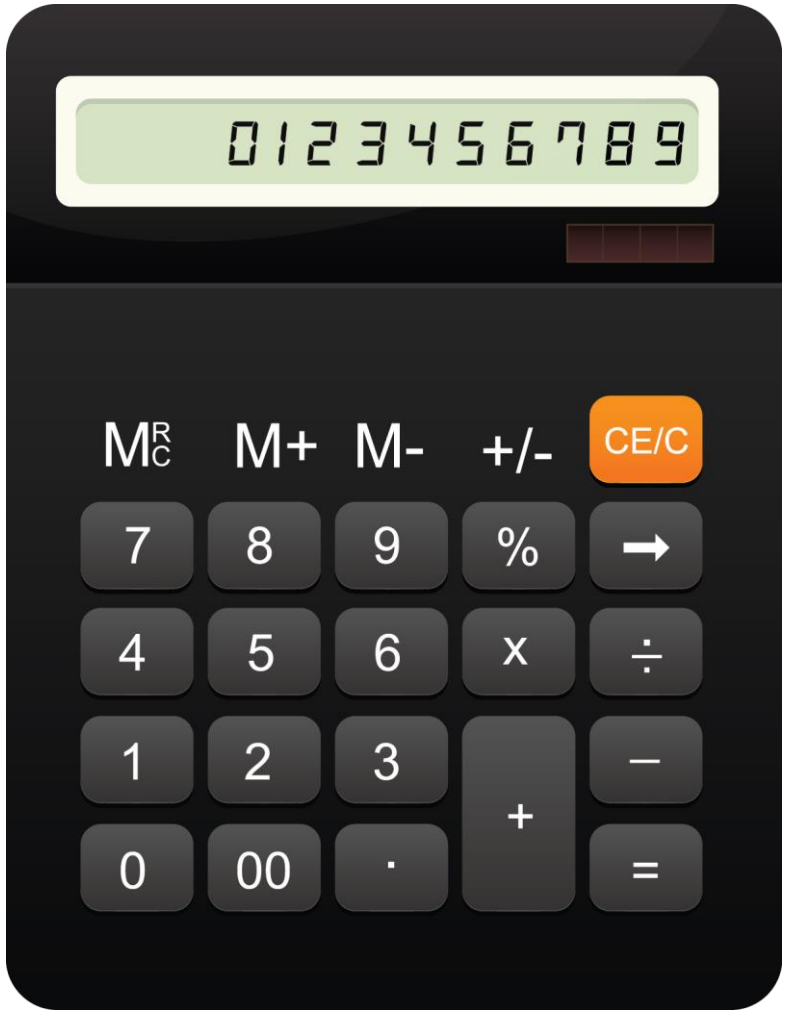
Type	Getter	Setter
Read-Only Attribute	O	X
Write-Only Attribute	X	O
Read-Write Attribute	O	O

ลักษณะการเข้าถึงข้อมูลที่สามารถกำหนดรูปแบบได้เป็นหนึ่งในคุณลักษณะของ OO เรียกว่า Encapsulation

CALCULATOR

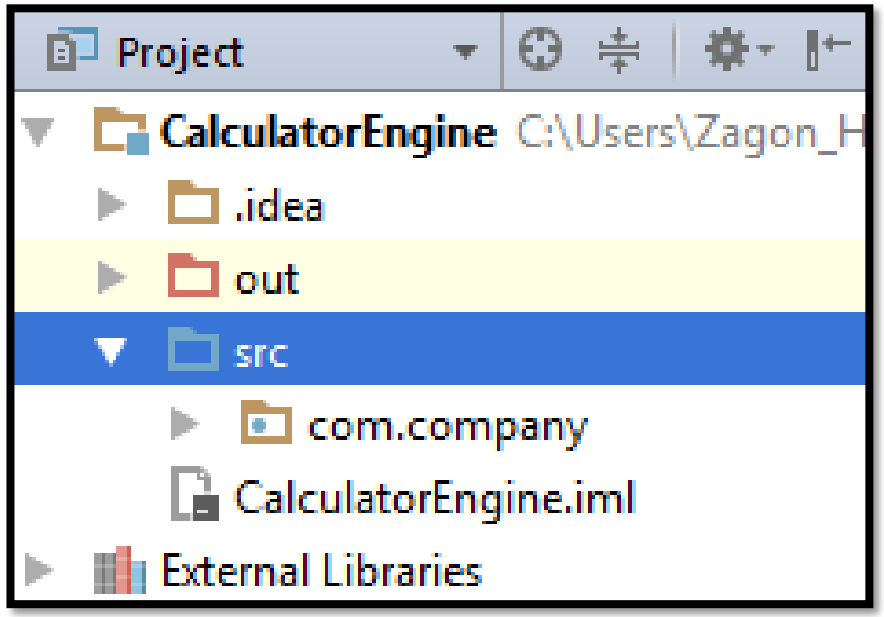
กรณีศึกษา เครื่องคิดเลข

- ถ้าเราต้องการคำตอบจากการคำนวณ 13-12 ต้องกดเครื่องคิดเลขแบบนี้



CALCULATOR

สร้าง **Project** ใหม่ ชื่อ `CalculatorEngine`



CALCULATOR

- ในการคำนวนนี้จะจำลองกระบวน โดยไม่สนใจ User Interface
- การพัฒนาซอฟต์แวร์ในแนวคิดของวิศวกรรมซอฟต์แวร์ในส่วนของการออกแบบและพัฒนากลไก **(mechanisms)** ก่อนสนใจเรื่อง User Interface เรียกว่า **abstraction**

CALCULATOR

แนวคิดของเครื่องคิดเลข

เครื่องคิดเลขเมื่อพิจารณาแล้วจะประกอบด้วย 2 ส่วน

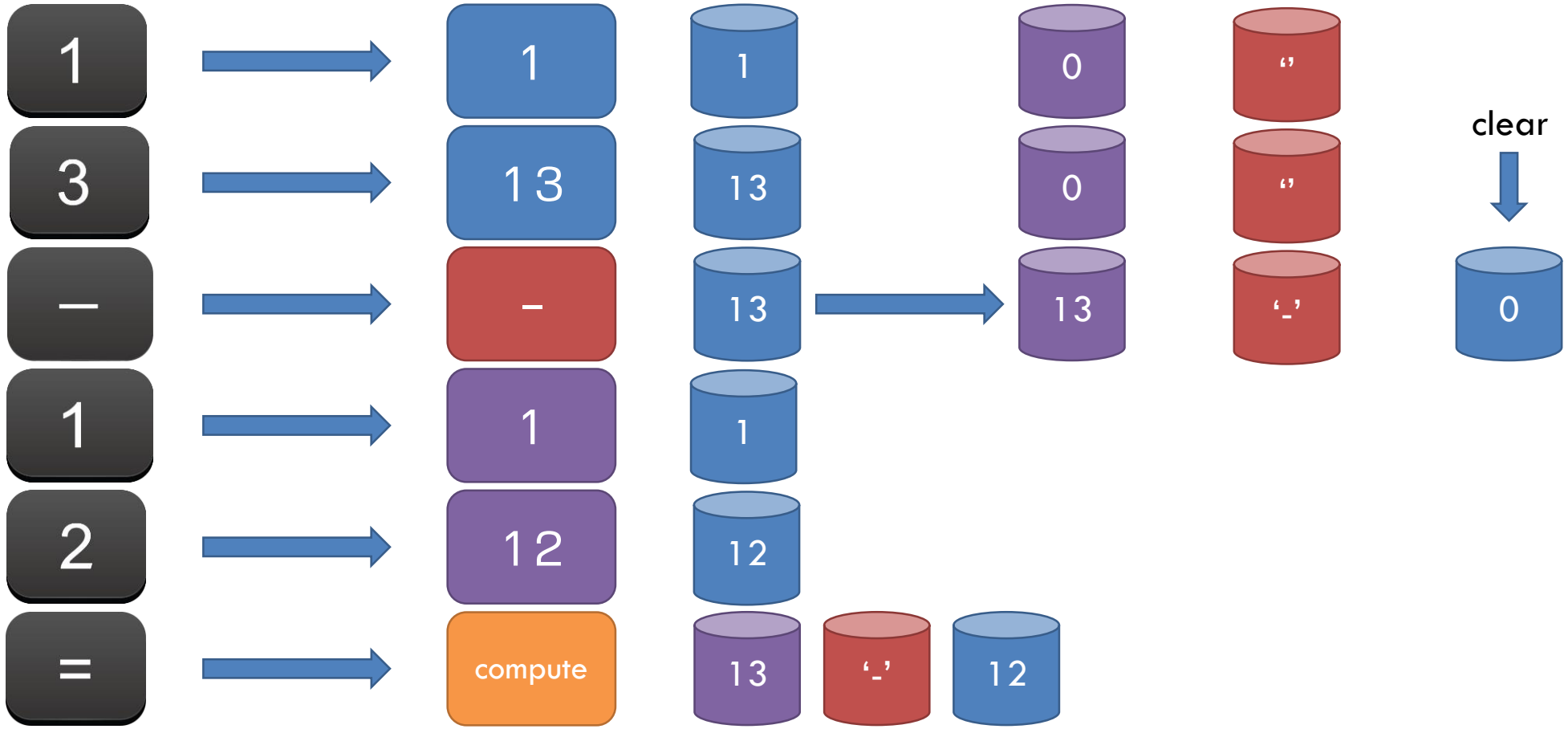
- ส่วนสำหรับ Values
- ส่วนสำหรับ Operator

เครื่องคิดเลขจะทำงานได้จะต้องประกอบไปด้วย 2 ส่วนนี้

CALCULATOR

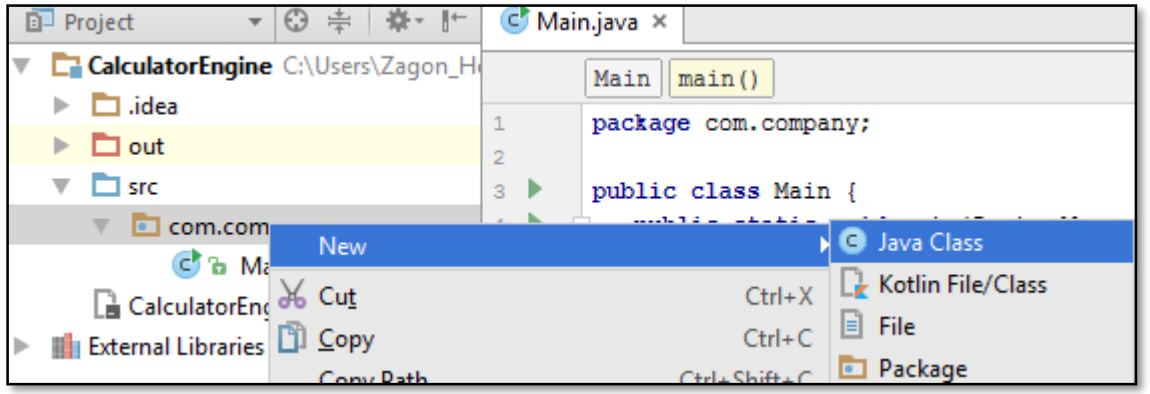
- ตัวดำเนินการ + - * / บนเครื่องคิดเลขหมายถึงต้องมี 4 **method** ในการคำนวณ
- **Method compute** ถูกกำหนดให้เป็น **method** ที่ใช้ในการคำนวณ
- **Method clear** ใช้ในการ **clear** ค่าของเครื่องคิดเลขเพื่อเริ่มกาคำนวณใหม่

CALCULATOR วิเคราะห์การทำงาน เริ่มจาก



CALCULATOR

- การออกแบบโครงสร้างเครื่องคิดเลขเริ่มจาก สร้าง **class CalculatorEngine**



CALCULATOR > digit(int x)

```
package com.company;

public class CalculatorEngine {
    int value;
    int keep;       // two calculator registers
    void digit(int x)       { ... }
}
```

โครงสร้างพื้นฐานคือ

- **int value** คือตัวแปรที่รับค่าชุดแรก
- **int keep** คือตัวแปรที่ไว้พักค่าจากชุดแรก
- **void digit (int x)** เป็น **method** ที่มารับค่าตัวเลข
  - Input 1 : value = 1
  - Input 1,2 : value = 12
  - Input 1,2,7 : value = 127

CALCULATOR > digit(int x)

- ปัญหาคือตอนนี้ไม่มี **method** สำหรับดูผลลัพธ์(แสดงตัวแปร **value**)
- เพิ่ม **method showValue** ไว้ดูค่า **int value**

```
package com.company;

public class CalculatorEngine {
    int value;
    int keep;        // two calculator registers
    void digit(int x)       { ... }
    int showValue(){
        return  value;
    }
}
```

CALCULATOR > digit(int x)

เริ่ม implement ได้

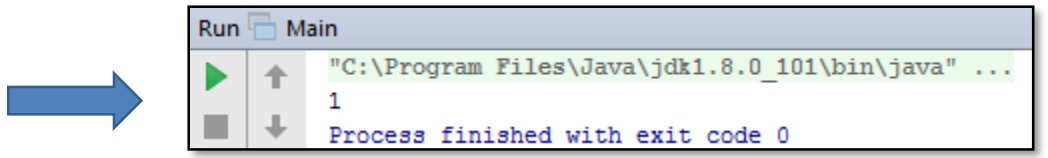
การดูผลลัพธ์ ต้องใช้ method main สร้าง object แล้วดูค่าดังนี้

```
package com.company;

public class Main {
    public static void main(String[] args) {
        CalculatorEngine ce = new CalculatorEngine();
        ce.digit(5);
        System.out.print(ce.showValue());
    }
}
```

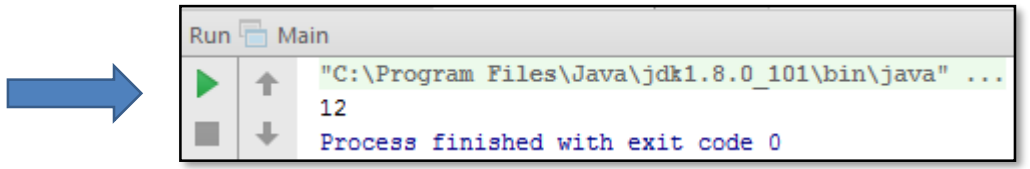
CALCULATOR > digit(int x)

ผลลัพธ์ที่ต้องการ

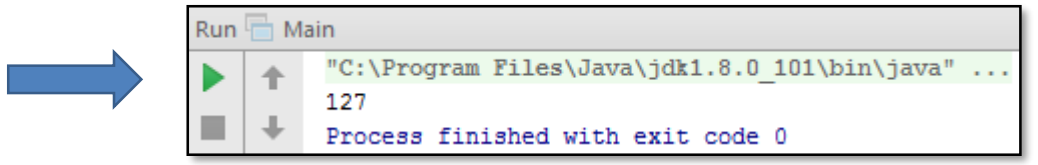
```
    public static void main(String[] args) {
        CalculatorEngine ce = new CalculatorEngine();
        ce.digit(1);
        System.out.print(ce.showValue());
    }
```




```
public static void main(String[] args) {
    CalculatorEngine ce = new CalculatorEngine();
    ce.digit(1);
    ce.digit(2);
    System.out.print(ce.showValue());
}
```




```
public static void main(String[] args) {
    CalculatorEngine ce = new CalculatorEngine();
    ce.digit(1);
    ce.digit(2);
    ce.digit(7);
    System.out.print(ce.showValue());
}
```

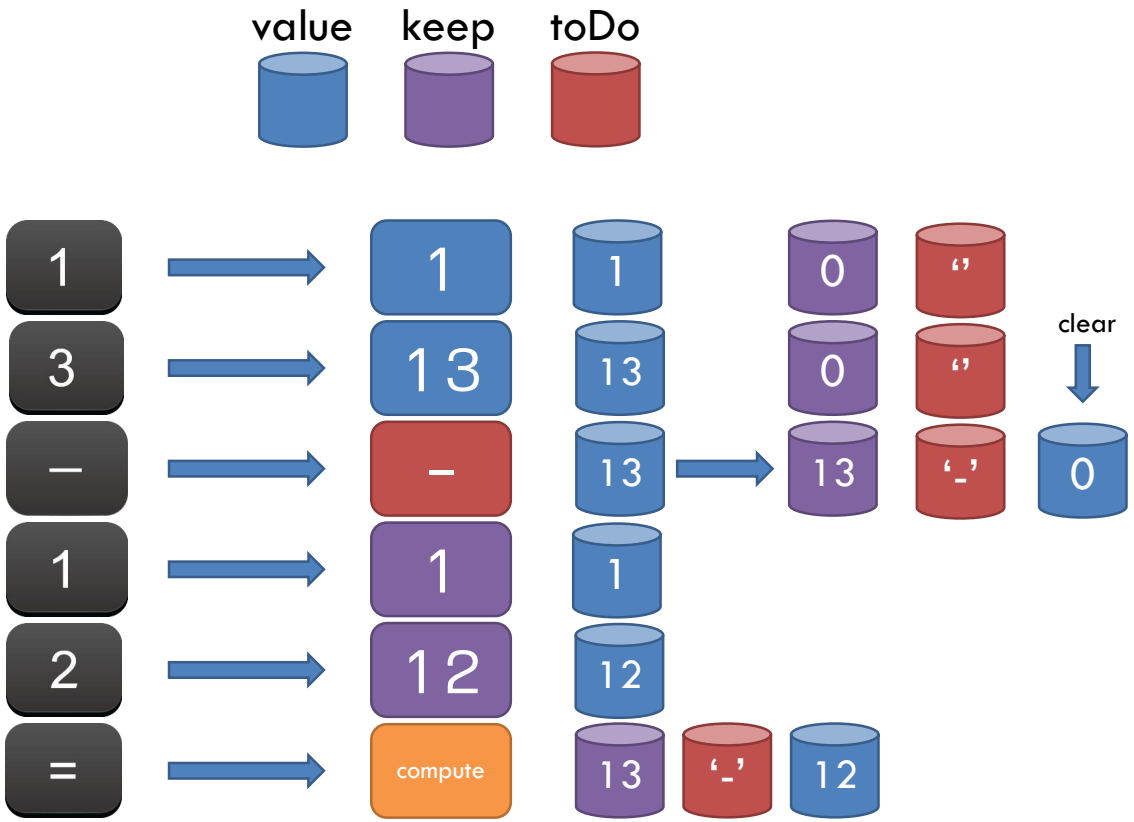


CALCULATOR > Operator Methods

Method ที่เป็นตัวดำเนินการประกอบด้วย

- add
- subtract
- multiply
- divide

CALCULATOR > Operator Methods



- subtract
  - keep = value
  - toDo = ‘-’
  - value = 0
- add
  - keep = value
  - toDo = ‘+’
  - value = 0
- multiply
  - keep = value
  - toDo = ‘*’
  - value = 0
- divide
  - keep = value
  - toDo = ‘/’
  - value = 0

CALCULATOR > Operator Methods

method void add() จะมีหน้าตาแบบนี้

```
void add(){
     keep = value;
     value = 0;
     toDo = '+';
}
```

ให้นักศึกษา implement

- void subtract()
- void multiply()
- void divide()

CALCULATOR > Operator Methods

- จะเห็นว่าทุก **method** ทำงานเหมือนกัน ต่างกันที่ **toDo**
- เกิดความซ้ำซ้อน
- แก้ไขได้โดยสร้าง **method** มาทำหน้าที่แทน แต่รับ **parameter** เป็นค่าของ **toDo**

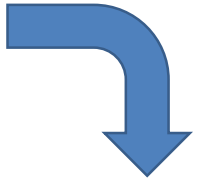
- subtract
  - keep = value
  - toDo = '-'
  - value = 0
- multiply
  - keep = value
  - toDo = '*'
  - value = 0
- add
  - keep = value
  - toDo = '+'
  - value = 0
- divide
  - keep = value
  - toDo = '/'
  - value = 0

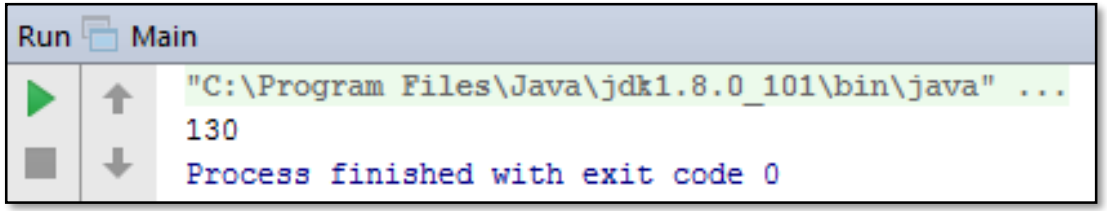
CALCULATOR > compute

- ออกแบบ **method** void compute() เพื่อใช้ในการคำนวน
- ต้องการให้ทำการคำนวณโดย
  - เช็ค **toDo** ว่าเป็น **operator** ใด
  - คำนวณแล้วเก็บผลลัพธ์ไว้ที่ **value**

CALCULATOR

```
public static void main(String[] args) {
    CalculatorEngine ce = new CalculatorEngine();
    ce.digit(1);
    ce.digit(2);
    ce.digit(7);
    ce.add();
    ce.digit(3);
    ce.compute();
    System.out.print(ce.showValue());
}
```





สรุป