Big Java Chapter 2 : Using Objects

2.1 Types

- ทุกๆค่า(value) ในjava จะต้องมี type เป็นของตัวเอง ex. 13 มีtype=int, “Hello” มีtype= String, System.out มีtype=PrintStream

- type เป็นตัวบอกว่าเราสามารถทำอะไรกับ value นี้ได้บ้าง ex. เราสามารถนำ int มาบวกกันได้, สามารถเรียก(call) println จากobj.ของPrintStreamได้

- เมื่อมี value เช่น 13 ,1.3 ปรากฏขึ้นมาใน java จะเรียกตรงที่ปรากฏขึ้นมาว่า ***number literal***

- ไม่ต้องใส่คอมม่าในตัวเลข ex. ~~13,000~~ -> 13000

- สำหรับเลขฐานสิบ $1.3x10^{-4}$ -> 1.3E-4

- ข้อมูลประเภท integer มีข้อได้เปรียบกว่า floating-pointตรงที่ ใช้พื้นที่น้อยกว่า, ประมวลผลได้เร็วกว่า, ไม่ต้องมีปัดหาเรื่องการปัดเศษ ดังนั้นควรใช้กับข้อมูลที่คิดว่าจะไม่มีจุดทศนิยมเด็ดขาด เช่น ความยาวของ array

- number type พวกนี้จัดเป็น primitive type ซึ่งหมายความว่า Number ไม่ได้เป็น Obj. ดังนั้นจะไม่สามารถใช้ method กับพวกมันได้ เช่น ~~13.toString()~~ แต่ว่ายังสามารถใช้ operator ต่างๆ เช่น + - * / กับพวกมันได้

- การรวมกันของ variables, literals, operators และ method เรียกว่า ***expression***

ex. x + y * 2

2.2 Variables

- การจะเก็บ value ไว้สำหรับใช้ทีหลัง จะต้องเก็บไว้ใน ***variable***

- variable จะประกอบด้วย type, name, value

  ex. String greeting = “Hello World”;

  String = type, greeting = name, “Hello World” = value

- variable จะถูกเรียกใช้ในตำแหน่งที่ต้องการใช้ value

  ex. System.out.print(greeting) เหมือนกับ System.out.print(“Hello World”)

- type ของ variable จะขึ้นอยู่กับ value ที่จะกำหนดให้ เช่น

ถ้าvalueเป็น10 typeก็ต้องเป็น int จะเป็น String ไม่ได้

<u>กฎการตั้งชื่อ</u> ฝ่าฝืนไม่ได้

- ชื่อตัวแปรจะประกอบด้วย ตัวอักษร, ตัวเลข, underscore(_), dollar sign($)

- ห้ามขึ้นต้นด้วยตัวเลข

- ห้ามมีช่องว่าง และสัญลักษณ์อื่นๆ

- ห้ามใช้คำสงวน เช่น public, new, void (คำที่พิมพ์ไปล่ะขึ้นสีอ่ะ)

มารยาทในการตั้งชื่อ ฝ่าฝืนได้

- การตั้งชื่อตัวแปรควรให้สื่อความหมายและสัมพันธ์กับข้อมูลที่จะเก็บ
- ชื่อ variable และ method ต้องขึ้นต้นด้วยตัวเล็ก และใช้ camel case (ex. getName, setId)
- ชื่อ class ต้องขึ้นต้นด้วยตัวใหญ่
- ไม่ควรมี $ ในชื่อ

Syntax : Variable Declaration

```
typeName varName = value;
typeName varName;
```

2.3 The Assignment Operator

- เมื่อต้องการแก้ไข้ value ของ variable ให้ใช้ = (assignment operator)
- variable ที่ยังไม่เคย assign ค่า ถ้าเอาไปใช้จะได้ error “uninitialized variable”
- การทำงานของ = คือการแทนที่ค่าเดิมของ variable
- width = width+10 หมายความว่า ให้นำ widthไปบวกกับ10 แล้วนำไปเก็บไว้ที่ width

Syntax : Assignment

```
varName = value;
```

2.4 Object, Class and Methods

- object คือค่า(value)ชนิดหนึ่ง ซึ่งสามารถจัดการกับข้อมูล(data)ภายในได้ โดยการเรียก(calling) method ต่างๆ
- method สามารถเข้าถึงข้อมูลต่างๆภายในobj.ได้ และผู้เรียกจะไม่รู้ว่า method นั้นทำงานอย่างไร (black box) เช่น “Hello”.length() เราไม่รู้ว่า เมธอด lengthทำงานยังไง แต่ก็ได้ค่าออกมา
- เมื่อ method ถูกเรียก กิจกรรมบ้างอย่างจะเกิดขึ้นมา โดยที่เราไม่รู้ว่ามันเกิดอะไรขึ้นบ้าง แต่ก็ได้ผลลัพธ์ออกมา เช่นเรียก .length() เราไม่รู้ว่ามันทำอะไร แต่ก็ได้ความยาวออกมา
- ประเภทของ obj. แต่ล่ะตัวคือ ***class***
- class แต่ล่ะclass จะมี method ที่เฉพาะสำหรับ object ของตัวเอง เช่น คลาสPrintStream จะมีเมธอด print และ println, คลาสString มีเมธอด length และ replace เป็นต้น
- เมื่อจะใช้method ต้องแน่ใจว่า method นั้นประกาศอยู่ในคลาสของ obj.นั้นจริงๆ
- สรุป ทุกๆobj.เป็นของclass และ classก็จะมีmethodเตรียมไว้สำหรับobj.ของตัวเอง
- ส่วนที่สามารถเข้าถึงได้ และสามารถบอกได้ว่าเราสามารถทำอะไรกับ obj. นั้นได้บ้าง เรียกว่า ***public interface***
- ส่วนที่ไม่สามารถเข้าถึงได้ ทั้งข้อมูล และการทำงานต่างๆซึ่งถูกซ่อนไว้ เรียกว่า ***private implementation***
- method ที่มีชื่อเดียวกัน อยู่ในclassเดียวกัน แต่รับ parameter ต่างกัน เรียกว่า ***overloaded***

2.5 Method Parameters and Return Values

ex. greeter.replace(“H”, “h”);

- เมื่อ method ถูกเรียก ค่าที่ถูกส่งผ่านเข้าไปในวงเล็บ จะเรียกว่า **explicit parameter** จากตัวอย่างก็จะเป็น “H” และ “h”

- เมื่อ method ถูกเรียก obj.ตัวที่เรียกmethodนั้น จะเรียกว่า **implicit parameter** จากตัวอย่างก็จะเป็น greeter

- method นั้นสามารถดึง data ต่างๆที่ถูกเก็บไว้ใน implicit parameter(object ที่เรียก) มาใช้ในการทำงานได้

- method บางตัวจะมี ***return value*** ซึ่งก็คือค่าที่ได้จากการเรียกใช้ method นั้นนั่นเอง

- return value สามารถนำไปใช้เป็น parameter ของอีก method หนึ่งได้ เช่น System.out.println(greeter.length())

Syntax : Method Declare

```
accessType returnType methodName (paraType1 paraName1, ...){
        statement;
}
```

ex. public int length(){ ... }

- ประเภทของ implicit parameter จะเป็นชนิดเดียวกับclassที่ประกาศ method นั้น

2.6 Constructing Objects

- ก่อนจะพูดในบทนี้ขอแนะนำให้รู้จักกับ คลาส Rectangle เพราะจะใช้อธิบายในบทนี้

- คลาสนี้จะเก็บข้อมูล4ตัวคือ x y (พิกัดมุมซ้ายบนของสี่เหลี่ยม) width height

- การจะสร้าง obj. ใหม่ของคลาส Rectangle จะต้องเรียก operator ***new***

ex. new Rectangle(5, 10, 20, 30)

- new จะทำการสร้าง Rectangle obj.

- ในวงเล็บจะส่งค่า 5, 10 ,20, 30 ให้เพื่อใช้ในการ **เซ็ทค่าเริ่มต้นให้กับ obj.**

Syntax : Object Construction

```
new ClassName(parameters)
```

- การทำงานที่สร้าง obj. เรียกว่า ***construction***

- parameter ที่ส่งเข้าไปเรียกว่า ***construction parameters***

-

2.7 Accessor and Mutator Methods

- เมธอดที่เข้าถึงข้อมูลใน obj และ return ข้อมูลนั้นออกมา โดยปราศจากการแก้ไขข้อมูลใดๆของ obj. เรียกเมธอดประเภทนี้ว่า ***accessor method***

- ส่วนเมธอดที่เข้าไปแก้ไขข้อมูลใน obj. เรียกว่า ***mutator method***

2.8 The API Documentation

- รายชื่อ/ข้อมูล คลาสและเมธอดต่างๆในจาว่า จะถูกเก็บรวบรวมไว้ใน ***API Document***

- API ย่อมาจาก Application Programming Interface

- โปรแกรมเมอร์ที่นำคลาสต่างๆเหล่านี้มาสร้างเป็นโปรแกรม เรียกว่า ***application programmer***

- โปรแกรมเมอร์ที่ทำหน้าที่สร้างคลาสต่างๆเหล่านี้ เรียกว่า ***system programmer***

- สามารถดู API Document ได้ที่ http://java.sun.com/java/7/docs/api/index.html

- API จะบอกว่า เมธอดต่างๆทำงานอะไร, มีพารามิเตอร์อะไรบ้าง, รีเทิร์นค่าอะไรออกมา

- คลาสที่มีความสัมพันธ์กัน จะถูกเก็บรวบรวมไว้ด้วยกันใน ***package***

- เช่น คลาสRectangle จะถูกเก็บไว้ใน package java.awt

Syntax : Importing a Class from a Package

```
import paclageName.ClassName;
```

ex. import java.awt.Rectangle;

- ตามปกติเวลาเราใช้คลาส String และ System เราไม่จำเป็นต้อง import เพราะอยู่ใน package ที่ชื่อว่า ***java.lang*** เพราะจะถูก import มาอัตโนมัติ

2.9 Implementing a Test Program

- ***test program*** เป็นโปรแกรมที่จะเรียก method อื่นมาเพื่อเช็คค่าที่returnออกมาว่ามีความถูกต้องหรือไม่

- การสร้าง test program ทำดังนี้

1. สร้าง tester class
2. สร้าง main method
3. construct object (สร้าง obj.)
4. เรียกใช้ method
5. แสดงผลที่ได้จาก method
6. แสดงผลที่ควรจะเป็น(ผลที่ถูกต้อง)

2.10 Object References

ex. Rectangle box = new Rectangle(0, 0, 10, 20);

- จากตัวอย่างที่สร้าง obj.Rectangle ใหม่ที่ชื่อ box แต่ในความเป็นจริงนั้น ตัวแปรboxไม่ได้เก็บ obj. เพียงแต่เก็บที่อยู่(***memory location***)ของ obj. ที่สร้างขึ้นมาใหม่เท่านั้น
- ที่ต้องทำแบบนี้เพราะว่า obj. นั้นมีขนาดใหญ่มาก การที่เก็บไว้เฉพาะที่อยู่ของobj. เป็นวิธีที่ดีกว่า
- การที่ variable เก็บที่อยู่แบบนี้ เรียกว่า ***Object reference***
- ทีนี้สมมติให้ Rectangle box2 = box;
- ทำแบบนี้จะทำให้ box และ box2 ชี้ไปที่ obj. ตัวเดียวกัน เวลาแก้ไขจะเป็นยังไงก็เหมือนตอน python อ่ะเนอะ
- อย่างไรก็ตาม ตัวแปรที่เป็นพวก **Number** นั้นเก็บค่าตัวเลขจริง เช่น int a = 13; aก็จะเก็บเลข13จริงๆ ไม่ได้ชี้ไปที่เลข13

2.11 Graphical Application and Frame Windows

- applicationที่มีการวาดรูปภายในwindow เรียกว่า ***graphical application***
- applicationแบบนี้มีความน่าใช้มากกว่า ***console application*** ที่มีแต่เพียงตัวหนังสือ
- กราฟฟิคต่างๆ จะแสดงภายในส่วนที่เรียกว่า ***frame***
- frame คือ จอwindow 1จอ ที่มี title bar

การสร้าง frame

```
import javax.swing.JFrame;


public class EmptyFrameViewer {
        public static void main(String[] args){
                JFrame frame = new JFrame();          // (1)
                frame.setSize(300,400);               // (2)
                frame.setTitle(“Aely”);               // (3)
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);   // (4)
                frame.setVisible(true)                // (5)
        }
}
```

อธิบายโค้ด

บรรทัดที่ (1) สร้าง obj. ชื่อ frame จากคลาส JFrame เพื่อเป็นหน้าจอที่จะใช้งาน

บรรทัดที่ (2) กำหนดความกว้าง(wide) และความสูง(height) ของ frame

บรรทัดที่ (3) กำหนดข้อความบน title bar

บรรทัดที่ (4) กำหนด “default close operation” ถ้าไม่มีบรรทัดนี้ เวลาปิด frame ถึงเฟรมจะหายไป แต่โปรแกรมยังคงทำงานอยู่

บรรทัดที่ (5) ทำให้ frame ปรากฏออกมา

2.12 Drawing on a Component

- การจะวาดอะไรลงไปใน frame เราไม่สามารถวาดใส่ลงไปดื้อๆได้ จะต้องสร้าง ***component*** obj. ขึ้นมาก่อน จากนั้นจึงใช้ฟังก์ชั่น add เพื่อเพิ่มcomponent นั้นลงไปใน frame

- ในเมื่อจะสร้าง obj. component แสดงว่าเราต้องสร้าง class component ก่อน แต่ว่า java มีคลาส JComponent มาให้ ซึ่งเราสามารถไป extends มาใช้งานได้เลย

Note : extent คือการสืบทอด เหมือนกับว่าสร้างคลาสลูกจากคลาส JComponent ที่มีอยู่แล้ว

```
public class RectangleComponent extends JComponent {
        public void paintComponent(Graphic g){
                //Drawing Instructions
        }
}
```

- จากตัวอย่างจะเห็นเมธอด paintComponent เมธอดนี้จะถูกเรียกโดยอัตโนมัติเมื่อ

1. window is resize (ไม่แปลน่าจะเข้าใจกว่า)
2. window ถูกแสดงหลังจากถูกซ่อน

และเมธอดนี้จะถูกเรียกผ่าน เมธอด add ของ JFrame

- เมธอดนี้จะรับพารามิตเตอร์ 1 ตัวคือ obj. ของคลาส Graphic

- Graphic เป็นคลาสที่มีอยู่แล้วในระบบ แต่มีเมธอดต่างๆไม่เพียงพอต่อการใช้งาน จึงควร convert g ที่เป็น obj. ของคลาสGraphic มาเป็น obj.ของคลาส Graphic2D แทน ซึ่งมีเมธอดให้ใช้งานมากกว่า

```
Graphic2D g2 = (Graphic2D) g;
```

Note : การ convert แบบนี้เรียกว่าวิธีการ cast ซึ่งจะได้เรียนในบทหลังๆ มั้ง

- การวาดรูปต่างๆสามารถทำได้โดยใช้เมธอด draw

```
Rectangle box = new Rectangle(5, 10, 30, 40);
g2.draw(box);
```

Applets

-

2.13 Ellipses, Line, Text and Color

- ไม่มีอะไรอ่ะ แค่สอนฟังก์ชั่น ไปดูๆเองล่ะกัน