Week2

Programming Fundamentals II

Assessment

- หมู่ 801,802 เรียนทฤษฎีด้วยกัน แต่แลปแยกวัน (พุธ,ศุกร์)
- จำนวนนิสิตที่เพิ่มขึ้น และ Fund สำคัญ
- สร้างวิธีการเรียนการสอนในหมู่เรียนนี้
  - ให้นิสิตเตรียมกระดาษ A4 ในการมาเรียนทุกครั้ง
  - ทบทวนเนื้อหา อาทิตย์ก่อนให้ 5-10นาที แรก
  - เข้าเรียนหลังจากนี้คือสาย ถ่ายรูป เก็บหลักฐานไว้
  - จะมีแบบฝึกหัด ไม่คิดคะแนนให้ลองทำ ท้ายคาบเอามาส่ง
  - บางอาทิตย์ จะมี Quiz เก็บคะแนนจะแจ้งต้นคาบ

Course Outline

1. P2J (Basic)
2. P2J (Control structures)
3. P2J (Collection types)
4. Classes and methods
5. Inheritance
6. Polymorphism
7. Interfaces
8. Testing and debugging
9. Events
10. UI programming
11. Exceptions
12. Generics
13. Concurrency
14. Team project

Week 1 **Data types, Variables,** Operators

- ตัวแปรนั้นต้องขึ้นต้นด้วยตัวอักษร
- ถัดจากตัวอักษรแรกของตัวแปรจะตามด้วยตัวอักษร หรือตัวเลข หรือเครื่องหมาย $ หรือเครื่องหมาย _ ก็ได้
- ตัวแปรในภาษา Java เป็น Case Sensitive นั่นคือ
- ห้ามตั้งชื่อตัวแปรที่ตรงกับคีย์เวิร์ด (Keyword) คำสงวน (Reserved Word) ในภาษา Java

```
dataType  VarName  = Value;
```

dataType	เป็นชนิดข้อมูลของตัวแปรที่ต้องการ
VarName	เป็นชื่อตัวแปรโดยตั้งตามกฎการตั้งชื่อ
Value	เป็นค่าของตัวแปร

Data types

เลขจำนวนเต็ม
1. int เป็น default
2. long ต้องระบุ l หรือ L หลังเลข

เลขทศนิยม
1. double เป็น default
2. float ต้องระบุ f หรือ F หลังเลข

Primitive Type	Size	Minimum Value	Maximum Value	Wrapper Type
char	16-bit	Unicode 0	Unicode 2^{16}-1	Character
byte	8-bit	-128	+127	Byte
short	16-bit	-2^{15} *(-32,768)*	$+2^{15}$-1 *(32,767)*	Short
int	32-bit	-2^{31} *(-2,147,483,648)*	$+2^{31}$-1 *(2,147,483,647)*	Integer
long	64-bit	-2^{63} *(-9,223,372,036,854,775,808)*	$+2^{63}$-1 *(9,223,372,036,854,775,807)*	Long
float	32-bit	32-bit IEEE 754 floating-point numbers		Float
double	64-bit	64-bit IEEE 754 floating-point numbers		Double
boolean	1-bit	`true` or `false`		Boolean
void	-----	-----	-----	Void

ตัวอย่างการตั้งชื่อตัวแปร

ชื่อตัวแปร	ถูกหรือผิด
X	✓
dayofWeek	✓
3dGraph	✖
Data1	✓
while	✖
Week day	✖

ลำดับความสำคัญของตัวดำเนินการ

ลำดับที่	ตัวดำเนินการ
1	( ), [ ]
2	++, --, !, ~
3	*, /, %
4	+, -
5	<<, >>, >>>
6	<, <=, >, >=
7	==, !=
8	&
9	^
10	|
11	&&
12	||
13	?:
14	=, +=, -=, *=, /=, %=, <<=, >>=, >>>=, &=, ^=, !=

ตัวอย่างการคำนวณของตัวดำเนินการ

1.) (2 + 3) * 5 = 25

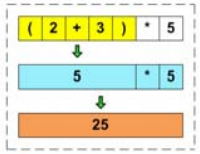
2.) 2 + 3 * 5

3. (7 + 3) * (10 - 2)

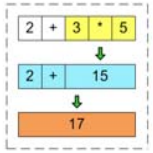
4. (5 + 2) * 15 % 4

ตัวอย่างการคำนวณของตัวดำเนินการ

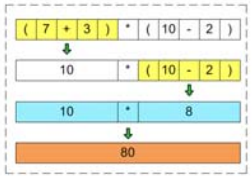
((2 + 3)1 * 5)2 = 25



(2 + (3 * 5)1)2



((7 + 3)1 * (10 - 2)2)3



(((5 + 2)1 * 15)2 % 4)3

Control Structures

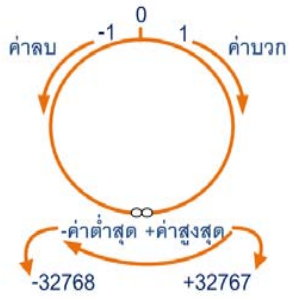
Programming Fundamentals II

Outline

1. Overflow + Declare constant
2. Sequence Control Statement
3. Selection Control Statement
4. Iteration Control Statement
5. Break
6. Continue
7. Condition expression
8. Static method

Overflow

การกำหนดตัวแปรเพื่อเก็บข้อมูลเลขจำนวนเต็มที่มากกว่า 32767 เป็นข้อมูลชนิด short integer ทำให้เกิดปัญหา overflow ของข้อมูล




```
1
2   public class IntegerRange {
3
4       public static void main(String[] args) {
5       short max = 32767;
6       System.out.println(max);
7       max++;
8       System.out.println(max);
9       max++;
10      System.out.println(max);
11    }
12  }
```

```
run:
32767
-32768
-32767
BUILD SUCCESSFUL (total time: 0 seconds)
```

ค่าคงที่ “Final”

- ค่าคงที่ (Constant) ทำหน้าที่เก็บข้อมูลที่ต้องการกำหนดไว้ไม่ให้มีการเปลี่ยนแปลงตลอดเวลา เช่น ค่า PI เป็นต้น
- การประกาศค่าคงที่เหมือนกับการประกาศตัวแปรที่มีการกำหนดค่าเพียงแต่จะมีการใช้คีย์เวิร์ด final นำหน้า

รูปแบบการใช้งาน

```
final  dataType  VarName  =  Value;
```

โดยที่ dataType เป็นชนิดข้อมูลของค่าคงที่ที่ต้องการ
VarName เป็นชื่อตัวแปรโดยตั้งตามกฎการตั้งชื่อ
Value เป็นค่าของค่าคงที่

ทิศทางการทำงานของโปรแกรม

ใช้คำสั่งควบคุมซึ่งมี 3 รูปแบบ คือ

1. คำสั่งแบบตามลำดับ (Sequence Control Statement)
2. คำสั่งแบบมีทางเลือก (Selection Control Statement)
   (if / else / else if / switch)
3. คำสั่งแบบทำซ้ำ (Iteration Control Statement)
   (while / do_while / for)

คำสั่ง break

คำสั่ง continue

Block Statement

- You can put a sequence of zero or more statements inside a pair of braces
- Form:

```
{
    variable-declarations
    class-declarations
    statements
}
```

- It is treated as a single statement
- Variables declared inside a block have a block scope

คำสั่งแบบตามลำดับ (Sequence Control Statement)

- ทุกคำสั่งจะทำงานตามลำดับ โดยไม่มีการข้ามคำสั่งใด
- แต่ละคำสั่งจะถูกเรียกใช้งานเพียงครั้งเดียว
- การทำงานเริ่มจากคำสั่งที่ 1 แล้วทำคำสั่งที่ 2 และถัดไปเรื่อยๆ จนครบ


```
public static void main(String[] args)
{
    // Start
    System.out.println("Start");

    // work2
    System.out.println("Work1..");

    // work3
    System.out.println("Work2...");

    // work4
    System.out.println("Work3....");

    // work5
    System.out.println("Work4......");

    // Stop
    System.out.println("Stop");
}
```

คำสั่งแบบมีทางเลือก (Selection Control Statement)

(if / else / else if / switch)

- Form 1:

```
if (condition)
    truebranch
```

- Form 2:

```
if (condition)
    truebranch
else
    falsebranch
```


(if / else / else if)

Ex: Shopping

```
if (money > 100) {
    System.out.println("Buy a sock.");
    money -= 50;
}

if (money > 80) {
    System.out.println("Buy a new game.");
    money -= 75;
} else {
    System.out.println("Save the rest.");
}
```


if

- ควบคุมให้โปรแกรมตัดสินใจทำงานหรือไม่ทำงานในชุดคำสั่งที่กำหนด
- ตรวจสอบจากนิพจน์ที่กำหนดว่าเป็นจริงหรือเท็จ

  ถ้านิพจน์ เป็นจริง (true) โปรแกรมจะทำงานที่ชุดคำสั่งที่อยู่ภายใต้คำสั่ง if

  ถ้านิพจน์ เป็นเท็จ (false) โปรแกรมจะข้ามไปทำงานที่คำสั่งต่อไปทันที

โดยที่

boolean_expression เป็นนิพจน์เงื่อนไข ซึ่งมีผลการตรวจสอบเป็น true หรือ false

statements เป็นชุดของคำสั่งที่จะทำงานเมื่อเงื่อนไขที่กำหนดให้เป็น true

Ex:

```
// work1
int i = 1;
if((true) && ( i == 1 ))
{
    System.out.println("If is true");
}
```


else

- ควบคุมให้โปรแกรมเลือกทำงานในชุดคำสั่งใดชุดคำสั่งหนึ่งจาก 2 ทางเลือก
- ตรวจสอบจากนิพจน์ที่กำหนดว่าเป็นจริงหรือเท็จ

  ถ้านิพจน์ เป็นจริง (true) โปรแกรมจะทำงานที่ชุดคำสั่งที่อยู่ภายใต้คำสั่ง if

  ถ้านิพจน์ เป็นเท็จ (false)โปรแกรมจะทำงานที่ชุดคำสั่งที่อยู่ภายใต้คำสั่ง else

โดยที่

boolean_expression เป็นนิพจน์เงื่อนไข ซึ่งมีผลการตรวจสอบเป็น true หรือ false

statements_1 เป็นชุดของคำสั่งที่จะทำงานเมื่อเงื่อนไขที่กำหนดให้เป็น true

statements_2 เป็นชุดของคำสั่งที่จะทำงานเมื่อเงื่อนไขที่กำหนดให้เป็น false

Ex:

```
// work1
int i = 2;
if((true) && ( i == 1 ))
{
    System.out.println("If is true");
}
else
{
    System.out.println("If is false");
}
```

else if

- ควบคุมให้โปรแกรมเลือกทำงานในชุดคำสั่งใดชุดคำสั่งหนึ่งจากหลายทางเลือก
- แต่ละทางเลือกจะมีการกำหนดนิพจน์เงื่อนไขเพื่อให้โปรแกรมตรวจสอบนิพจน์

หากพบว่าทางเลือกไหนมีนิพจน์เป็นจริง (true) ก็จะทำงานที่ชุดคำสั่งภายในทางเลือกนั้น โดยไม่พิจารณาทางเลือกอื่นที่ยังไม่ได้ทำการตรวจสอบอีก มีรูปแบบการทำงาน ดังนี้

```
if   (boolean_expression_1)
{
   statement_1;
}
else  if (boolean_expression_2)
{
   statement_2;
}
Else
 {
   statement_n;
}
```

Ex:

```
// work2
int a = 3;
if( a == 1 )
{
    System.out.println("a = "+a);
}
else if ( a == 2)
{
    System.out.println("Else if a==2 ");
}
else
{
    System.out.println("Else");
}
```


ข้อควรระวัง if/ else

- ในกรณีที่มีคำสั่งมากกว่า 1 คำสั่ง และไม่ใส่เครื่องมือหมาย { } ครอบคำสั่งทั้งหมด จะทำให้โปรแกรมทำงานเฉพาะคำสั่งแรกเท่านั้น ทำให้ผลการทำงานผิดพลาดได้

```
if(a > b)
    System.out.println("a less than b");
else
    temp = a;
    a = b;
    b = a;
    System.out.println("a and b are swap");
```

```
if(a > b)
    System.out.println("a less than b");
else{
    temp = a;
}
    a = b;
    b = a;
    System.out.println("a and b are swap");
```

- ควรใช้งานคำสั่ง if โดยกำหนดคำสั่งภายใต้เครื่องหมาย { } เสมอ

```
if (a < b)
  System.out.println("a less than b");
else {
  temp = a;
  a = b;
  b = a;
  System.out.println("a and b are swap");
}
```


Switch

- Form:

```
switch (expression) {
    case constant1: branch1
    case constant2: branch2
    …
    default: branchn
}
```

- The *expression* must evaluate to one of these types:
  - an int, short, char, or byte
  - a boxed version of the above
  - an enum
  - a String

Switch

- ควบคุมให้โปรแกรมเลือกทำงานในชุดคำสั่งใดชุดคำสั่งหนึ่งจากหลายทางเลือก
- แต่ละทางเลือกจะมีการกำหนดเงื่อนไขของแต่ละทาง โดยตรวจสอบเงื่อนไขแต่ละทางเลือก
- หากพบว่าทางเลือกใดมีเงื่อนไงเป็นจริง (true) จะทำงานที่ชุดคำสั่งภายในทางเลือกนั้น
- โดยไม่พิจารณาทางเลือกอื่นที่ยังไม่ตรวจสอบอีก มีรูปแบบการทำงานดังนี้

```
switch (expression)
{
  case list_value_1:  statement_1;
                      break;
  case list_value_2:  statement_2;
                      break;
  case list_value_n:  statement_n;
                      break;
 default : statement;
}
```

Ex:

```
int key = 2;

System.out.println("Start...\n\n");

switch(key)
{
    case 1:
        System.out.println("Count One: "+key);
        break;
    case 2:
        System.out.println("Count Two: "+key);
        break;
    case 3:
        System.out.println("Count Tree: "+key);
        break;
    default :
        System.out.println("Default...");
}
```

P2J if/ else if/ switch

Python

```
if day == "Sunday":
    action = "Do the gardening."
elif day == "Monday":
    action = "Go to the office."
elif day == "Friday":
    action = "Hang out with friends."
elif day == "Saturday":
    action = "Play the piano."
else:
    action = "Work, work, and work."
```


P2J if/ else if/ switch

JAVA

```
if (day.equals("Sunday"))
    action = "Do the gardening.";
else if (day.equals("Monday"))
    action = "Go to the office.";
else if (day.equals("Friday"))
    action = "Hang out with friends.";
else if (day.equals("Saturday"))
    action = "Play the piano.";
else
    action = "Work, work, and work.";
```


P2J if/ else if/ switch

JAVA

```
switch (day) {
    case "Sunday":
        action = "Do the gardening.";
        break;
    case "Monday":
        action = "Go to the office.";
        break;
    case "Friday":
        action = "Hang out with friends.";
        break;
    case "Saturday":
        action = "Play the piano.";
        break;
    default:
        action = "Work, work, and work.";
}
```


ทิศทางการทำงานของโปรแกรม

ใช้คำสั่งควบคุมซึ่งมี 3 รูปแบบ คือ

1. คำสั่งแบบตามลำดับ (Sequence Control Statement)
2. คำสั่งแบบมีทางเลือก (Selection Control Statement)
   (if / else / else if / switch)
3. **คำสั่งแบบทำซ้ำ (Iteration Control Statement)**
   (while / do_while / for)

The `while` statement

- Form:
```
while (condition)
    body
```


The `do-while` statement

- Form:
```
do
    body
while (condition);
```

- Equivalent to:
```
body
while (condition)
    body
```


while

- มีการตรวจสอบเงื่อนไขที่เป็นนิพจน์ทางตรรกศาสตร์ก่อนการทำงานชุดคำสั่งภายในลูปทุกครั้ง
- โปรแกรมจะทำงานที่ชุดคำสั่งภายในลูปก็ต่อเมื่อนิพจน์ที่กำหนดเป็นจริง
- เมื่อโปรแกรมทำงานที่ชุดคำสั่งภายในลูปเสร็จสิ้น โปรแกรมจะตรวจสอบนิพจน์ใหม่อีกครั้ง จะหยุดการทำซ้ำเมื่อนิพจน์ที่กำหนดเป็นเท็จ มีรูปแบบการทำงานดังนี้

```
while (bolean_expression)
{
    statements;
}
```

Ex:

```
int i=1;

while( i <= 30)
{
    if( i == 5)
    {
        break;
    }

    System.out.println("Run..." +i);

    i++;
}
```

โดยที่

boolean_expression	เป็นนิพจน์ที่เป็นเงื่อนไขทางตรรกศาสตร์มีการตรวจสอบก่อนการทำงานชุดคำสั่งภายในลูปทุกครั้ง
statements	เป็นชุดคำสั่งที่ต้องการให้มีการทำซ้ำ

do while

- มีการทำงานในชุดคำสั่งอย่างน้อย 1 รอบ ก่อนตรวจสอบเงื่อนไขที่เป็นนิพจน์ทางตรรกศาสตร์หลังคำสั่ง while
- ถ้านิพจน์ที่กำหนดเป็นจริง จะทำงานชุดคำสั่งภายในลูปอีกครั้ง ถ้านิพจน์ที่กำหนดเป็นเท็จโปรแกรมจะออกจากลูปการทำงานไปทำงานที่คำสั่งถัดไปทันที มีรูปแบบการทำงานดังนี้

```
do
{
    statements;
} while (bolean_expression);
```

Ex:

```
int i = 5;
System.out.println("Start\n\n");

do{
    System.out.println("FalseLoop i = "+i);
}while(false);

do{
    System.out.println("Loop i = "+i);
    i++;
}while(i <=5);
```

โดยที่

statements	เป็นชุดของคำสั่งที่ต้องการให้มีการทำซ้ำ
boolean_expression	เป็นนิพจน์เงื่อนไขทางตรรกศาสตร์ที่มีการตรวจสอบหลังทำงานในชุดคำสั่งเสร็จสิ้นทุกครั้ง

The `for` statement

- Form:

```
for (initialization; condition; step)
    body
```

- Equivalent to:

```
initialization
while (condition) {
    body
    step
}
```

- Example:

```
for (int i = 0; i < 10; i++)
    System.out.println("Counting " + i + "...");
```


for

- มีการตรวจสอบเงื่อนไขที่เป็นนิพจน์ทางตรรกศาสตร์ก่อนการทำงานที่ชุดคำสั่งภายในลูปทุกครั้ง
- โดยโปรแกรมจะทำงานที่ชุดคำสั่งภายในลูปเมื่อเงื่อนไขเป็นจริง ซึ่งจะมีการเพิ่มหรือลดค่าตัวแปร

มีรูปแบบการทำงานดังนี้

```
for (control_variable = value; boolean_expression; increment or decrement)
{
    statements;
}
```

โดยที่

control variable	เป็นตัวแปรที่ควบคุมจำนวนครั้งของการทำซ้ำ มีชนิดเป็นจำนวนเต็ม
value	เป็นค่าเริ่มต้นที่กำหนดให้ตัวแปรควบคุม
boolean expression	เป็นนิพจน์เงื่อนไขทางตรรกศาสตร์ที่มีการตรวจสอบเงื่อนไข
increment or decrement	เป็นคำสั่งกำหนดการเพิ่มค่าหรือลดค่าของตัวแปรควบคุม
statements	เป็นชุดของคำสั่งที่ต้องการให้มีการทำซ้ำ

P2J for Python

```
for i in range(4):
    for j in range(i+1):
        print "*",
    print
```


P2J for Java

```
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < i + 1; j++)
        System.out.print("*");
    System.out.println();
}
```

P2J for

```
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < i + 1; j++)
        System.out.print("*");
    System.out.println();
}
```

=

```
int i = 0; // This is not exactly equivalent
while (i < 4) {
    int j = 0; // Neither this one (but closer)
    while (j < i + 1) {
        System.out.print("*");
        j++;
    }
    System.out.println();
    i++;
}
```


ทิศทางการทำงานของโปรแกรม

ใช้คำสั่งควบคุมซึ่งมี 3 รูปแบบ คือ

1. คำสั่งแบบตามลำดับ (Sequence Control Statement)
2. คำสั่งแบบมีทางเลือก (Selection Control Statement)
   (if / else / else if / switch)
3. คำสั่งแบบทำซ้ำ (Iteration Control Statement)
   (while / do_while / for)

คำสั่ง break
คำสั่ง continue

break

- เป็นคำสั่งที่ใช้ร่วมกับคำสั่งการทำซ้ำ เพื่อหยุดการทำงานก่อนครบตามจำนวนรอบที่กำหนด
- หรือใช้เป็นคำสั่งหยุดการทำงาน **switch** เพื่อข้ามคำสั่งอื่นๆ ที่อยู่ภายในบล็อกของ **switch**

```
import java.util.Scanner;
public class Break_Test {
  public static void main(String args[]) {
    int i = 1, score = 100;
    Scanner scan = new Scanner(System.in);
    while (i <= 5) {
      System.out.print("ป้อนข้อมูลคะแนนนิสิตคนที่ " + i + " >>> ");
      score  = scan.nextInt();
      if ((score > 100) || (score < 0)) {
        System.out.println(score + " เป็นข้อมูลคะแนนที่ไม่ถูกต้อง");
        break;
      }
      System.out.println(score + " เป็นข้อมูลคะแนนที่ถูกต้อง");
      i = i+1;
    }
  }
}
```

```
run:
ป้อนข้อมูลคะแนนนิสิตคนที่ 1 >>> 95
95 เป็นข้อมูลคะแนนที่ถูกต้อง
ป้อนข้อมูลคะแนนนิสิตคนที่ 2 >>> 80
80 เป็นข้อมูลคะแนนที่ถูกต้อง
ป้อนข้อมูลคะแนนนิสิตคนที่ 3 >>> 90
90 เป็นข้อมูลคะแนนที่ถูกต้อง
ป้อนข้อมูลคะแนนนิสิตคนที่ 4 >>> 101
101 เป็นข้อมูลคะแนนที่ไม่ถูกต้อง
BUILD SUCCESSFUL (total time: 16 seconds)
```


continue

- เป็นคำสั่งที่ทำงานตรงข้ามกับคำสั่ง break
- จะเป็นคำสั่งที่บังคับให้โปรแกรมข้ามไปทำงานในรอบต่อไปทันที โดยไม่สนใจคำสั่งที่เหลืออยู่ในลูป

```
import java.util.Scanner;
public class Continue_Test {
  public static void main(String args[]) {
    int i = 1, score = 100;
    Scanner scan = new Scanner(System.in);
    while (i <= 5) {
      System.out.print("ป้อนข้อมูลคะแนนนิสิตคนที่ " +
              i + " >>> ");
      score  = scan.nextInt();
      if ((score > 100) || (score < 0)) {
        System.out.println(score +
                " เป็นข้อมูลคะแนนที่ไม่ถูกต้อง");
        continue;
      }
        System.out.println(score +
                " เป็นข้อมูลคะแนนที่ถูกต้อง");
        i = i+1;
    }
  }
}
```

```
run:
ป้อนข้อมูลคะแนนนิสิตคนที่ 1 >>> 100
100 เป็นข้อมูลคะแนนที่ถูกต้อง
ป้อนข้อมูลคะแนนนิสิตคนที่ 2 >>> 90
90 เป็นข้อมูลคะแนนที่ถูกต้อง
ป้อนข้อมูลคะแนนนิสิตคนที่ 3 >>> 85
85 เป็นข้อมูลคะแนนที่ถูกต้อง
ป้อนข้อมูลคะแนนนิสิตคนที่ 4 >>> 101
101 เป็นข้อมูลคะแนนที่ไม่ถูกต้อง
ป้อนข้อมูลคะแนนนิสิตคนที่ 4 >>> 100
100 เป็นข้อมูลคะแนนที่ถูกต้อง
ป้อนข้อมูลคะแนนนิสิตคนที่ 5 >>> 80
80 เป็นข้อมูลคะแนนที่ถูกต้อง
BUILD SUCCESSFUL (total time: 24 seconds)
```

Condition expression

- Form:
  e1 ? *e2* : *e3*
- The expression evaluates to *e2* if *e1* is true, and to *e3* otherwise
- Example:

```
// Obtain the absolute value of x
y = x >= 0 ? x : -x;
```


Static method

- Simple form (use this form for now):

```
public static return-type name(formal-list)
  method-body
```

- Static methods belong to a class and can be called without an object

Static Example

```
class StaticMethodTest {
    public static int abs(int n) {
        return n >= 0 ? n : -n;
    }

    public static void main(String[] args) {
        int n = (int)(11 * Math.random()) - 5;
        System.out.println("A random positive value " + abs(n));
   }
}
```


End.... Week2

Programming Fundamentals II

Lap1:
- เริ่มต้นภาษา Java
- JAVA: Data type, Variable, Operator
- JAVA: Control Structures

1.1 ติดตั้งเครื่องมือที่ใช้ในการสร้างโปรแกรม JAVA (Eclipse)

- การติดตั้ง Eclipse เนื่องจากตัวโปรแกรม Eclipse เองเป็น Java-based ดังนั้นก่อนที่เราจะสามารถติดตั้ง Eclipse ได้จำเป็นต้องติดตั้ง Java development Kit (JDK) เสียก่อน (ปัจจุบัน 1/17 Java SE Development Kit 8u121)
- ทำการดาวน์โหลดโปรแกรม Eclipse เลือก Eclipse Neon 4.6.2 โดยคลิ๊กเลือกลือกรุ่น 32 Bit หรือ 64 Bit และ OS ตามที่ใช้งาน
- เมื่อเปิดเข้า Eclipse ครั้งแรกจะต้องกำหนด Workspace หรือที่บันทึกไฟล์งานของเรา ก็เลือกกำหนดตามที่เราต้องการ ในที่นี้ให้ตั้ง folder ใน Drive ที่เก็บข้อมูลเป็น Workspace ที่จะบันทึกไฟล์งาน
- เปิดใช้งานครั้งแรกของโปรแกรม Eclipse จะเปิดหน้า Welcome เป็นอันเสร็จสิ้นขั้นตอนการลงโปรแกรม Eclipse ที่พร้อมจะใช้งานได้ต่อไป

1.2 เริ่มเขียนภาษา JAVA และทดลองสั่งงาน

- สร้าง Java project โดยตั้งชื่อ Project name ว่า “Lab01_รหัสนิสิตXXXXX” คลิ๊ก Finish ถ้ามีหน้าต่าง pop-up ให้ตอบตกลงไปก่อน
- ใน Package Explorer จะมี Project ขึ้นมา ให้นิสิต คลิ๊กขวาที่โฟลเดอร์ src > new > class ให้ตั้งชื่อ class ว่า Lab1SimpleClass เป็นอันเสร็จสิ้นขั้นตอนการสร้าง class พื้นฐาน

ให้นิสิตเพิ่มข้อความเข้าไปใน class Lab1SimpleClass ดังนี้

```
public class Lab1SimpleClass
{
	public static void main(String[] args)
	{
		System.out.println("18 and 81: " + 24 + 45);
	}
}
```

ให้ save file แล้วทำการสั่ง Run จะได้ผลเช่นไร

- เปลี่ยนข้อความใน Method main เป็น `System.out.println("18 and 81: " + (24 + 45));`
ให้ save file แล้วทำการสั่ง Run จะได้ผลเช่นไร

- ให้นิสิต คลิ๊กขวาที่โฟลเดอร์ src > new > class ให้ตั้งชื่อ class ว่า Lab2SimpleClass จากนั้นให้นิสิตสร้าง Method main และพิมพ์ข้อความว่า sysout และให้ทดลองกด ctrl + spacebar คำสั่ง println จะถูกเรียกใช้ได้ทันที ให้นิสิตพิมพ์ข้อความใส่ใน input ของ method println ตามตารางและให้แสดงผลลัพท์ใส่ในตาราง

ชุดคำสั่ง	ผลลัพท์
System.out.println("I Love JAVA");	
System.out.println("I Love 'JAVA'");	
System.out.println("I Love "JAVA"");	
System.out.println("I Love \JAVA\");	
System.out.println("I Love \\JAVA\\");	
System.out.println("I Love \"JAVA\"");	
System.out.println("\tI Love JAVA");	
System.out.println("I Love JAVA,OOP");	

1.3 ชนิดข้อมูล, ตัวแปร, ตัวดำเนินการ

- ให้นิสิต คลิ๊กขวาที่โฟลเดอร์ src > new > class ให้ตั้งชื่อ class ว่า Lab3SimpleClass จากนั้นให้นิสิตสร้าง Method main และประกาศตัวแปร int ดังข้อความด่านล่าง

```
public class Lab3SimpleClass
{
	public static void main(String[] args)
	{
		int keys = 88;
		System.out.println("A piano has " + keys +  keys);

	}
}
```

ให้ save file แล้วทำการสั่ง Run จะได้ผลเช่นไร

- เปลี่ยนข้อความเป็น `System.out.println("A piano has " + keys + " keys");`
ให้ save file แล้วทำการสั่ง Run จะได้ผลเช่นไร

- ให้นิสิต คลิ๊กขวาที่โฟลเดอร์ src > new > class ให้ตั้งชื่อ class ว่า Lab4SimpleClass จากนั้นให้นิสิตสร้าง Method main และประกาศตัวแปร int ดังข้อความด่านล่าง

```
public class Lab4SimpleClass {

	public static void main(String[] args)
	{
		int a, b, c;
		a = 4;
		b = 7;
		c = a * b;
		System.out.println(a + " x " + b + " = " + c );
	}
}
```

ให้ save file แล้วทำการสั่ง Run จะได้ผลเช่นไร

- จากความรู้จากการสร้าง Class ตั้งแต่ Lab1SimpleClass ถึง Lab4SimpleClass ให้นิสิตเขียนโปรแกรมเพื่อแสดงผลลัพท์ดังต่อไปนี้ ออกทาง console และให้ตั้งชื่อ class ว่า Lab5SimpleClass

ผลลัพท์ที่คาดหวัง 4 x 8 – (8 + 4) = ผลคำนวณของ 4 x 8 – (8 + 4) คือ 20

เขียน Code ทั้งหมดใน class Lab5SimpleClass ลงที่ใส่ข้อความข้างล่าง

- ให้นิสิตเขียนโปรแกรมแปลงองศา Celsius เป็นองศา Fahrenheit โดยให้ ป้อนค่าตัวแปร Celsius = 24 และให้ตั้งค่าตัวแปรคงที่ในโปรแกรม .ให้ตั้งชื่อ Class ว่า Lab6CelToFar
การตั้งค่าตัวแปรคงที่โปรแกรม ให้ประกาศ final หน้า Data type เช่น final int BASE = 32;

Hint: $Fahrenheit = \frac{9}{5} Celsius + 32$ ผลลัพท์ที่คาดหวัง Celsius Temperature: 24 Fahrenheit Equivalent: ผลคำนวณของโปรแกรม

เขียน Code ทั้งหมดใน class Lab6CelToFar ลงที่ใส่ข้อความข้างล่าง

1.4 แบบทดสอบ

ให้นิสิตเขียนคลาสชื่อ AddRandom ซึ่งสุ่มค่าทศนิยมมา 2 จำนวนในช่วง 0.0-100.0 และแสดงค่าทั้งสองพร้อมผลบวก

ตัวอย่างผลลัพท์ (หมายเหตุ การรันแต่ละครั้งจะให้ผลลัพธ์ที่ต่างกัน)

Sum of 13.515746408089147 + 99.32580969308835 = 112.84155610117749

การสุ่มค่าใน Java จะใช้คลาส Random โดยมีขั้นตอนการใช้ดังนี้

1. import คลาส Random ที่ต้นไฟล์

   เขียนคำสั่งนี้ที่ต้นไฟล์

   import java.util.*;

2. ประกาศตัวแปร random จากคลาส Random

   ใส่คำสั่งนี้ใน Method main

   Random random = new Random();

3. สุ่มค่ามาใช้

   เช่น สุ่มค่ามาเก็บไว้ในตัวแปร a สมมติว่าต้องการผลลัพธ์เป็นชนิด double

   a = random.nextDouble();

   random.nextDouble() จะให้ผลลัพธ์เป็นค่าสุ่มในช่วง 0.0-1.0 เก็บไว้ใน a แต่เราต้องแปลงช่วงให้เป็นช่วง 0.0-100.0 ตามข้อกำหนดของโจทย์

อ้างอิง

Java Random number generator https://docs.oracle.com/javase/8/docs/api/java/util/Random.html