Lanjutan Control Flow Sa-training.

Format

```
If (Expression1) {
   Executed-block1;
} else if (Expression2) {
   Executed-block2;
} Else {
   Executed-block3;
}.
```

→ Jika hasil Evaluasi Expression1 = true maka Executed_Block1 akan dijalankan.
→ Jika hasil Evaluasi Expression 1 = False maka Expression 2 akan dijalankan.
→ Jika hasil Evaluasi Expression 2 = true maka Executed_block2 akan dijalankan.
→ Jika hasil Evaluasi Expression 2 = False maka Executed-block 3 akan dijalankan.

Format

```
switch (argument) {  /* argumen dari switch terbatas */
   case condition_1 : Executed_case_1.
                      Break;
   case condition_2 : Executed_case_2
                      Break;
   .
   case condition_n : Executed_case_n
                      Break;
   default          : Executed_default
                      Break;
```

→ Jika argument == condition_1 maka Executed_case1 akan dijalanca
→ Jika argument == condition 2 maka Executed_case2 akan dijalankan
→ Jika argument == condition n maka Executed_casen akan dijalankan
→ Jika argument tidak sama dengan salah satu condition_?, maka Executed_default akan dijalankan.

Format

```
While (expression) {
   Executed_Block;
}
```

→ Jika hasil Evaluasi Expression = True maka Executedblock akan dijalankan.
→ Jika hasil Evaluasi Expression = False maka Excutedblock tidak akan dijalankan.

```
do {
   Executed_block;
} while(expression);
```

→ Jika hasil evaluasi Expression = true maka Executedblock akan dijalankan.
→ Jika hasil evaluasi Expression = False maka Executedblock tidak akan dijalankan.
→ karena Evaluasi Expression dilakukan dibawah, maka Executed_Block setidaknya akan dijalankan 1 kali

Format

```
for (initialization ; expression ; increment) {
   Executed-block;
}
```

→ Bagian initialization hanya akan dijalankan 1 kali pada waktu for(;;) pertama kali dijalankan.
→ Setelah bagian initialization dijalankan, maka bagian Expression akan di Evaluasi.
→ Jika hasil Evaluasi Expression = true, maka Executed_block akan dijalankan.
→ Jika hasil Evaluasi Expression = false, maka proses for(;;) berhenti
→ Setelah bagian Executed block maka bagian increment akan dijalankan
→ Setelah bagian increment dijalankan, maka bagian Expression akan di Evaluasi, seperti itu seterusnya sampai hasil Expression = false.

*Array adalah Wadah (Container) yang menampung sejumlah elemen data yg memiliki type yang sama.

- Declaration

{Data type}[] {Variable Name};

- Instantiation

{Variable name} = new {Data type} [{Size of Array}];

- Declaration, Instantiation, and initialization

{Data type}[] {Variable name} = {Value-1, Value-2, ..., Value-n};

- Accessing

{Variable Name}[{Index of Array}];

{Variable Name}[{Index of Array}] = {Some of Value};

*Primitive & class variable.
Variabel non array hanya dapat menampung 1 nilai.

String actorName = "Chris Pratt";
Int Age = 48;

* Array Primitive

int[] numberOfDaysInMonth = {31, 28, 31, 30, dst};

*Array of class

String[] nameOfDays = {"Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"};

04 - Agustus - 2023. SA_Training ~~Op~~ Penggabungan Operator conditional dan Comparison.

```
int bilA = 5, BilB : 5;
int bilC : 10, bilD : 11;
boolean boolAnd1, boolAnd2;
boolean boolOr1, boolOr2;
```

```
boolAnd1 : BilA < BilB && bilC < bilD;
boolAnd2 : bilA <= bilB && bilC < bilD;
boolOr1 : bilA < bilB || bilC < BilD;
boolOr2 : bilA < bilB || bilC == bilD;
```

boolAnd1 : 5 < 5 && 10 < 11
: False && True
: False.

boolAnd2 : 5 <= 5 && 10 < 11
: True && True.
: True.

Bool OR1 = 5 < 5 || 10 < 11
= false || True.
: True.

Bool OR2 = 5 < 5 || 10 == 11
= false || false
: false.

operator	Tipe Data	Penjelasan
~~&&~~	boolean	memberikan hasil true jika kedua Operand bernilai True, selain itu memberikan hasil false.
||	boolean.	memberikan hasil false jikaa kedua Operand bernilai false, selain itu memberikan hasil True.
? :	boolean	Menjalankan salah satu bagian jika hasil expression true, atau menjalankan bagian lain. jika hasil Expressian false.

Operator Assignment

```
int intA = 5,
    intB = 6,
    intC = 7,
    intD = 8,
    intE = 9;
```

```
intA += 2;
intB -= 2;
intC *= 2;
intD /= 2;
intE %= 2;
```

intA += 2
intA = intA + 2
= 5 + 2
= 7

intB -= 2
intB = intB - 2
= 6 - 2
= 4.

intC *= 2
intC = intC x 2
= 7 x 2
= 14.

intD /= 2
intD = intD / 2
= 8 / 2
= 4

intE %= 2
intE = intE % 2
= 9 % 2
= 1.

Input from keyboard

Import the class

```
import java.util.Scanner;
```

Declare Variable

```
Scanner scanner = new Scanner(System.in);
```

Read from keybord

```
String inputData = Scanner.nextLine();
```

Control Flow :

Statement dalam suatu source code normalnya di eksekusi dari urutan paling atas sampai urutan paling bawah.

Control Flow statement memungkinkan eksekusi statement dari source code tidak mengikuti aturan normal yaitu dari urutan paling atas ke urutan paling bawah.

*Decision Making Statement

-> IF
-> IF - else
-> IF - else IF - else
-> switch.

*Branching Statement

-> Break
-> Continue
-> return.

*. Looping Statement

-> While
-> Do - While
-> For.

Decision Making Statement If

* Format

```
If (expression){
   Executed_block;
}
```

-> Jika hasil evaluasi Expression = true maka Executed Block akan dieksekusi / dijalankan.

-> Jika hasil evaluasi Expression = false maka Executed-Block tidak akan dieksekusi.

Format

```
IF (expression){
   executed block-1;
}else {
   Executed block-2;
}
```

-> Jika hasil Evaluasi Expression = true, maka Executed Block 1 akan dieksekusi
-> Jika hasil Evaluasi Expression = false, maka Executed Block 2 akan dieksekusi.

```
/* Declaration and Initialization */;
Scanner scanner = new scanner (System.in);
String keyboardInput;
Int maxSpeed, CurrentSpeed;
/* Read data from keyboard */
System.out.printf ("Max Speed : ");
KeyboardInput = Scanner.NextLine();
maxSpeed = Integer.parseInt(KeyboardInput);
System.out.printf("Your Speed : ");
keyboardInput = Scanner.nextLine();
CurrentSpeed = Integer.parseInt (KeyboardInput);
```

```
/* Control Flow IF-ELSE */
IF(currentSpeed > maxSpeed){
System.out.printf("%n Be careful!, please slow your speed! %n");
}Else.{
System.out.printf(" %n Youare in.the. right speed. %n ");
}.
```

Class Constructor Sa-training

Format

modifier classname

```
public Airplane (){
    System.out.printf(" constructor() invoke ...");
}
```

modifier classname parameter

```
public Airplane (int value){
    System.out.printf(" constructor() invoke... ");
}
```

* Object

Creating Object

{Class name} {Variable Name} = New {Class Name};

Airplane airplane = New Airplane();

Activities when creating Object

- Variable declaration
- Instantiation
- Initialization.

Compile Executor

javac -d bin -sourcepath src src\namafile.java.

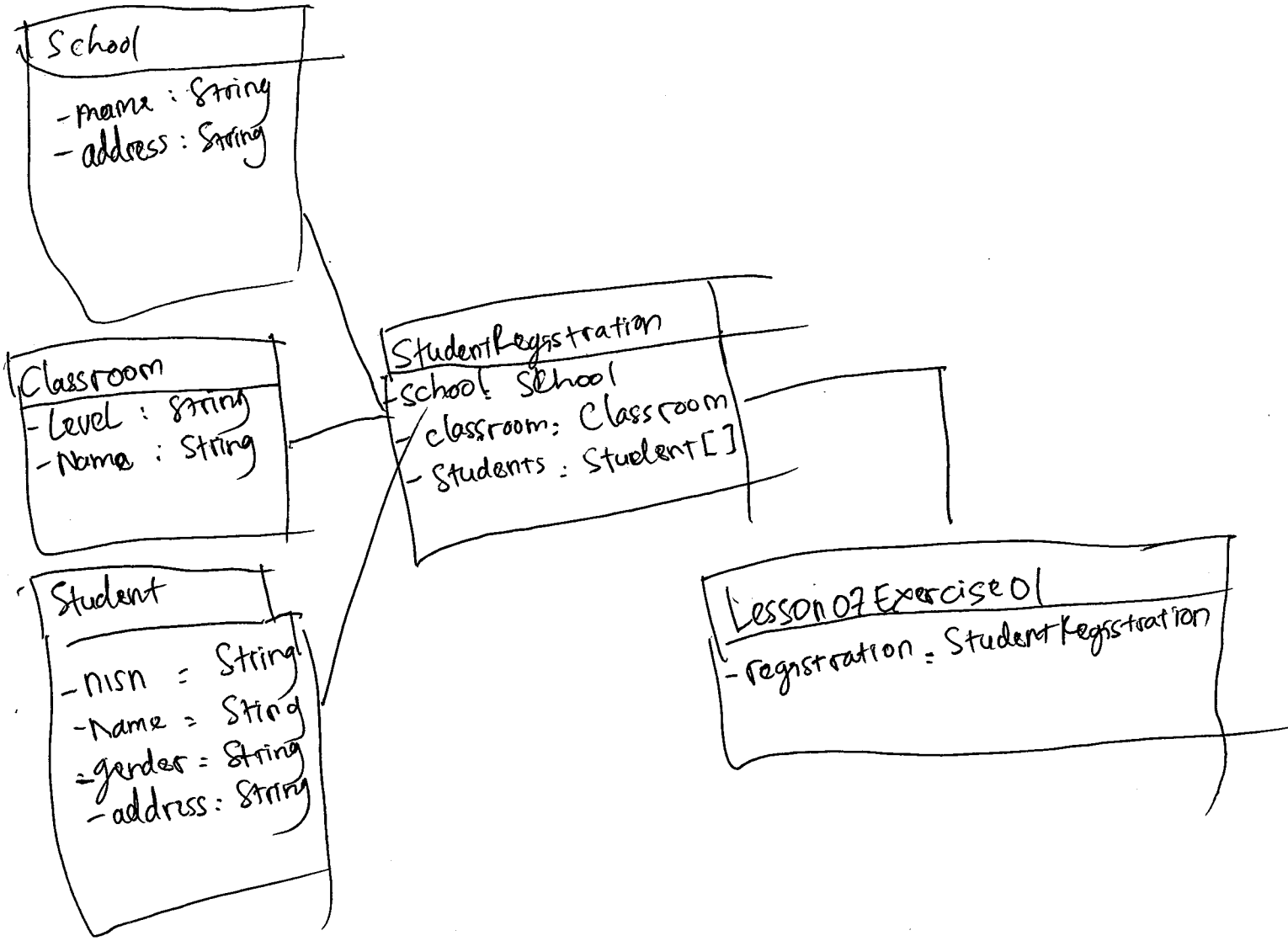
* Encapsulation

membungkus data beserta metod² yang akan mengakses data itu menjadi satu unit.

Class = Single unit

Field = data

Method = Operation.


School
-name : String
-address : String
Classroom
-Level : String
-Name : String
StudentRegistration
-school: School
-classroom: Classroom
-students : Student[]
Student
-nisn = String
-Name = String
-gender = String
-address: String
Lesson07Exercise01
-registration = StudentRegistration