BINUS UNIVERSITY ONLINE LEARNING

People
Innovation
Excellence

LECTURE NOTES

COMP6598

Introduction to Programming

Minggu 7

Sesi 11

Methods

People
Innovation
Excellence

LEARNING OUTCOMES

1. LO1 : Peserta diharapkan dapat merancang dan menerapkan algoritma yang tepat dalam menyelesaikan permasalahan

OUTLINE MATERI(Sub-Topic):

1. Defining a method
2. Calling a method
3. Void method
4. Passing parameter by value
5. Overloading method
6. The scope of variables
7. Exercise (Discussion forum 7 & Team Assignment 3)

ISI MATERI

Introduction

Method merupakan sekumpulan *statement* atau baris proses yang dikelompokkan menjadi satu bagian sehingga memungkinkan untuk dijalankan berkali-kali tanpa harus membuat proses yang sama per modulnya. Perhatikan contoh berikut ini

```
public class ComputeSum {
    public static void main(String[] args){
    int sum = 0;

    for (int i = 1; i <= 10; i++)
        sum += i;
    System.out.println("Sum from 1 to 10 is " + sum);

    sum = 0;
    for (int i = 20; i <= 37; i++)
    sum += i;
    System.out.println("Sum from 20 to 37 is " + sum);

    sum = 0;
    for (int i = 35; i <= 49; i++)
    sum += i;
    System.out.println("Sum from 35 to 49 is " + sum);
    }
}
```

Pada program ComputeSum diatas terlihat bahwa code yang sama harus diulang tiga kali dengan input yang berbeda yaitu penjumlahan dari 1 – 10, angka 20 – 37 dan angka 35 – 49. Program diatas dapat disederhanakan menggunakan method seperti pada program dibawah ini. Pengulangan perhitungan dikelompokan menjadi satu method yang bernama sum (baris 2 – 7) . Program di bawah ini dapat dipakai berkali kali dengan input yang berbeda beda.

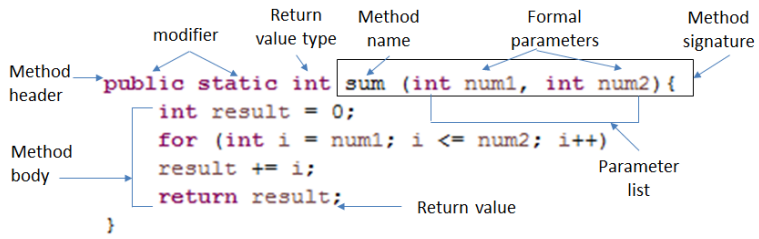
```
1  public class ComputeSum {
2      public static int sum (int num1, int num2){
3          int result = 0;
4          for (int i = num1; i <= num2; i++)
5          result += i;
6          return result;
7      }
8
9      public static void main(String[] args) {
10         System.out.println("Sum from 1 to 10 is " + sum(1,10));
11         System.out.println("Sum from 20 to 37 is " + sum(30,37)
12         System.out.println("Sum from 35 to 49 is " + sum(35,49)
13     }
14 }
```

Defining a Method

Sebuah defnisi method terdiri atas nama method, parameter, tipe return value, dan isi method. Syntax untuk mendefinisikan sebuah method adalah

```
modifier returnValueType methodName(list of parameters) {
// Method body;
}
```

Perhatikan contoh dibawah ini untuk mendefiniskan sebuah method.



Method header terdiri atas *modifier*, tipe *return value*, nama method dan daftar parameter. Sebuah method dapat mengembalikan nilai dan dapat juga tidak mengembalikan nilai. Apabila tidak mengembalikan nilai maka tipe return value adalah void.

Variabel yang didefinisikan pada method header adalah formal parameter atau lebih singkat disebut parameter. Saat method di panggil, maka nilai akan dikirimkan ke parameter, nilai ini didapatkan dari actual parameter atau argument. Nilai yang dikirimkan dari actual parameter harus sesuai dengan urutan yang ada pada formal parameter. Parameter bersifat optional.

Isi method atau method body terdiri dari sekumpulan statement yang mengerjakan fungsi dari method tersebut. Apabila method mengembalikan nilai, maka keyword return diperlukan.

Pada contoh di atas method bernama sum yang akan mengembalikan nilai dengan tipe int. method sum memiliki 2 parameter yaitu num1 dan num2. Pada method body, terdapat statements yang berfungsi untuk menghitung penjumlahan dari num1 sampai num2 yang

hasilnya disimpan di variabel result. Isi dari result menjadi return value dari method sum tersebut. Method berhenti saat statement return dijalankan.

Dalam beberapa bahasa programming, method dapat disamakan dengan istilah procedure dan function. Biasanya method yang mengembalikan nilai disebut function dan method yang tidak mengembalikan nilai disebut procedure.

Beberapa istilah seperti define dan declare sering digunakan. Definition menjelaskan item apa saja yang ada, sedangkan declaration biasanya merujuk kepada alokasi memori untuk menyimpan item data apa saja yang diperlukan.

Dari adanya penggunaan method terdapat beberapa keuntungan yakni sebagai berikut.

1. Mengurangi kode yang berulang
2. Memungkinkan penggunaan kembali kode yang sama tanpa perlu membuat kode yang sama
3. Meningkatkan kualitas struktur program
4. Memodulkan / mengelompokkan kode program
5. Membuat kode program menjadi lebih mudah untuk dibaca.
6. Mempersempit cakupan dalam proses *debbuging*.

Calling a Method

Calling a method (memanggil method) berarti menjalankan *code* yang ada pada method. Terdapat dua cara untuk memanggil sebuah method, tergantung dari apakah method tersebut mengembalikan nilai atau tidak. Apabila sebuah method mengembalikan sebuah nilai, maka pemanggilan method tersebut biasanya diperlakukan seperti sebuah nilai, seperti contoh berikut ini

```
int larger = max(3,4);
```

method max(3,4) diapnggil dan nilainya di assign ke variable larger. Contoh lainnya adalah dengan perintah

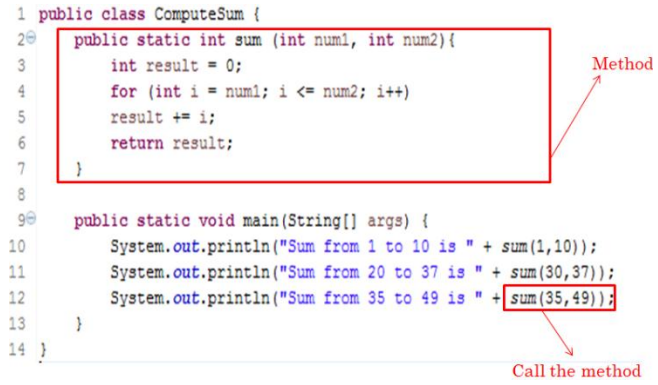
System.out.println(max(3,4));

Maka perintah diatas akan mencetak nilai yang dihasilkan oleh method max.

Jika sebuah method mengembalikan nilai void, maka pemanggilan atas method tersebut harus sebagai sebuah statement. Method yang mengembalikan nilai juga dapat dipanggil sebagai sebuah statement, dalam hal tersebut maka Java akan mengabaikan return value dari method tersebut.

Ketika sebuah program memanggil sebuah method, maka kendali atas program tersebut dipindahkan ke method yang dipanggil. Kendali akan dikembalikan Apabila menjumpai statement return ataupun sampai pada akhir method.

Perhatikan contoh di bawah ini :



- Program diatas terdiri atas 1 main modul dan 1 method dengan nama method sum (baris 2 – 7) yang berfungsi untuk menjumlahkan bilangan dari num1 sampai dengan num2
- Program akan dimulai dari main program, dimana method sum dipanggil sebanyak tiga kali (baris 10-12). Pada saat method sum dipanggil, maka dua nilai dikirimkan yaitu 1 dan 10, untuk pemanggilan sum(1,10). Nilai 30 dan 37 untuk pemanggilan sum(30,37). nilai 35 dan 49 untuk pemanggilan sum(35,49)
- Nilai yang dikirimkan akan masuk dalam parameter yang ada di method sum. Dimana nilai yang pertama di kirimkan akan menjadi nilai dari num1 dan nilai kedua menjadi nilai sum2.

- Method sum akan menjalankan statement yang berada di body method , dan nilai yang dihasilkan akan dikembalikan dengan keyword **return** (baris 6)
- Pada saat mencapai keyword return, maka kendali program akan dikembalikan ke main program dan akan mencetak nilai dari method sum tersebut.

Perhatikan contoh lainnya di bawah ini :

```
1  public class TestMax {
2      public static void main(String[] args) {
3          int i = 5;
4          int j = 2;
5          int k = max(i,j) ;
6          System.out.println("The maximum of " + i + " and " + j + " is " + k);
7      }
8
9      public static int max(int num1, int num2){
10         int result;
11         if (num1 > num2)
12             result = num1;
13         else
14             result = num2;
15         return result;
16     }
17 }
```

- Program TestMax terdiri atas 1 main modul dan 1 method dengan nama max.
- Program dimulai dari main, pada saat statement int x = max (i,j) , maka method max dipanggil dengan parameter yang ditransfer adalah i dan j. Hasil dari method tersebut akan dimasukkan sebagai nilai dari k.
- Saat method dipanggil , maka kendali program diserahkan kepada method.
- Method max menerima dua parameter yaitu num1 dan num 2, dimana num1 akan diisi dengan nilai dari variabel i dan num2 akan diisi dengan nilai dari variabel j.
- Setelah statement pada method dijalankan, maka nilai terbesar akan dikembalikan dengan keyword return.
- Pada saat program menjumpai keyword return maka kendali program dikembalikan ke main program yang selanjutnya akan menerima nilai return tersebut dan dimasukkan pada variabel k. statement selanjutnya adalah mencetak nilai dari i, j dan k

Void Method

Method dengan tipe void tidak mengembalikan nilai. Perhatikan contoh di bawah ini :

```
1  public class TestVoidMethod {
2      public static void main(String[] args) {
3          System.out.print("The grade is ");
4          printGrade(78.5);
5          System.out.print("The grade is ");
6          printGrade(59.5);
7      }
8
9      public static void printGrade(double score){
10         if (score >= 90.0) {
11             System.out.println('A');
12         }
13         else if (score >= 80.0) {
14             System.out.println('B');
15         }
16         else if (score >= 70.0) {
17             System.out.println('C');
18         }
19         else if (score >= 60.0) {
20             System.out.println('D');
21         }
22         else {
23             System.out.println('E');
24         }
25     }
26 }
```

- Program TestVoidMethod diatas memiliki 1 main dan 1 method yang bernama printGrade
- Saat main program memanggil method printGrade (78.5), maka kendali diserahkan kepada method tersebut.
- Method printGrade memiliki 1 parameter dengan tipe double. Nilai yang dikirimkan dari main akan menjadi nilai dari variabel score
- Method kemudian akan menampilkan nilai grade yang sesuai.
- Pada method void tidak terdapat keyword return

Passing Parameter by Values

Argument yang dikirimkan adalah berupa nilai ke parameter pada saat pemanggilan method. Saat pemanggilan method maka harus menyediakan argument yang memiliki urutan yang sama dengan parameter pada method signature. Hal ini dikenal dengan nama ***paremeter order association***.

Perhatikan contoh berikut :

```
1  public class Increment {
2      public static void main(String[] args) {
3          int x = 1;
4          System.out.println("Before the call, x is " + x);
5          System.out.println("After the call, x is " + x);
6          increment(x);
7          }
8
9      public static void increment(int n) {
10         n++;
11         System.out.println("n inside the method is " + n);
12     }
13 }
```

```
Before the call, x is 1
After the call, x is 1
n inside the method is 2
```

- Program Increment memiliki 1 main dan 1 method dengan nama increment. Walaupun nama program memiliki nama yang sama dengan method , hal ini dimungkinkan karena Java mengenai Case Sensitive, sehingga huruf besar dan huruf kecil dianggap berbeda.
- Main program akan memanggil method dengan statement increment(x), dimana method increment akan dipanggil dan nilai dari variabel x akan dikirimkan ke method increment dan dimasukan dalam variabel n pada method increment
- Method akan dijalankan dan mencetak nilai dari n++. Setelah program mencapai akhir dari method, maka kendali program dikembalikan ke main.

Perhatikan contoh di bawah ini:

```
1  public class TestPassByValue {
2      public static void main(String[] args) {
3          int num1 = 1;
4          int num2 = 2;
5          System.out.println("Before invoking the swap method, num1 is " +num1
6                  + " and num2 is " + num2);
7          swap(num1, num2);
8          System.out.println("After invoking the swap method, num1 is " + num1
9                  + " and num2 is " + num2);
10     }
11
12     public static void swap(int n1, int n2) {
13         System.out.println("\tInside the swap method");
14         System.out.println("\t\tBefore swapping, n1 is " + n1+ " and n2 is " + n2);
15         int temp = n1;
16         n1 = n2;
17         n2 = temp;
18         System.out.println("\t\tAfter swapping, n1 is " + n1 + " and n2 is " + n2);
19     }
20 }
```

```
Before invoking the swap method, num1 is 1 and num2 is 2
        Inside the swap method
                Before swapping, n1 is 1 and n2 is 2
                After swapping, n1 is 2 and n2 is 1
After invoking the swap method, num1 is 1 and num2 is 2
```

- Program di atas memiliki method yang bernama swap dengan dua parameter yaitu n1 dan n2.
- Program akan dimulai dari main, setelah statement pada baris 2 – 6 dijalankan, maka method swap dipanggil dengan statement swap(num1, nnum2)
- Saat method swap dipanggil, maka nilai num1 dan num2 akan dipassing ke n1 dan n2.
- Kemudian pada method swap dilakukan pertukaran pada n1 dan n2 (baris 15-16)
- Setelah itu n1 dan n2 dicetak.
- Kendali program dikembalikan ke main dan menjalankan statement selanjutnya di main (baris 9-10)
- Walaupun nilai n1 dan n2 di tukar, hal ini tidak akan mengubah nilai num1 dan num2 pada main.

Overloading Method

Method overloading memungkinkan programmer untuk membuat method dengan nama yang sama selama method signaturenya berbeda. Method overloading juga dapat membuat program lebih jelas dan mudah dibaca. Method yang menjalankan fungsi yang sama dengan

parameter yang berbeda seharusnya diberikan nama yang sama. Method overloading harus memiliki list parameter yang berbeda.

Perhatikan contoh di bawah ini :

```
1  public class TestMethodOverloading {
2      public static void main(String[] args) {
3          System.out.println("The maximum of 3 and 4 is "+ max(3, 4));
4          System.out.println("The maximum of 3.0 and 5.4 is "+max(3.0, 5.4));
5          System.out.println("The maximum of 3.0, 5.4, and 10.14 is "+max(3.0, 5.4, 10.14));
6      }
7
8      public static int max(int num1, int num2){
9          if (num1 > num2)
10             return num1;
11         else
12             return num2;
13     }
14
15     public static double max(double num1, double num2){
16         if (num1 > num2)
17             return num1;
18         else
19             return num2;
20     }
21     public static double max(double num1, double num2, double num3){
22         return max(max(num1, num2), num3);
23     }
24 }
```

```
The maximum of 3 and 4 is 4
The maximum of 3.0 and 5.4 is 5.4
The maximum of 3.0, 5.4, and 10.14 is 10.14
```

- Program TestMethodOverloading terdiri atas 1 main dan 3 method. Ketiga method tersebut memiliki nama yang sama namun method signature berbeda.
- Method max yang pertama memiliki list parameter 2 variabel dengan keduanya memiliki tipe integer (baris 8)
- Method max yang kedua memiliki list parameter 2 variabel dengan keudanya memiliki tipe double (baris 15)
- Method max yang ketiga memiliki list parameter 3 variabel dan ketiganya memiliki tipe double (baris 21).
- Pada saat main memanggil method max, maka method max yang dijalankan adalah sesuai dengan list parameter yang ada.
- Pada method max yang ketiga, di body method memanggil method max yang memiliki 2 parameter dengan tipe double sebanyak dua kali (baris 22).
- Pada baris ketiga saat max(3, 4) dijalankan maka ada kemungkinan dua method yang sama dijalankan yaitu max (double, double) dan max (int,int). Namun Java Compiler

akan menjalankan method yang paling spesifik, sehingga method yang dijalankan adalah max(int,int)

Ambiguous Invocation

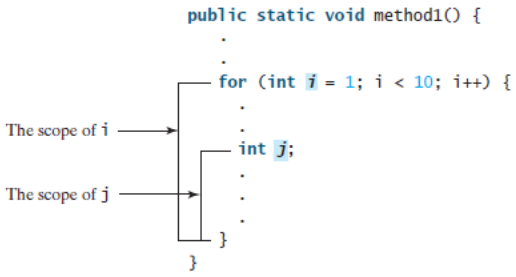
Terkadang ada kemungkinan method yang cocok lebih dari satu, tetapi compiler tidak dapat menemukan mana yang method yang sesuai. Hal ini dsebut sebagai ambiguous invocation dan dapat menyebabkan compile error. Seperti pada contoh berikut ini dimana terdapat dua method max dengan dua parameter yaitu integer dan double. Pada saat main memanggil max (1,2) maka terjadi ambiguous invocation ini, karena compiler tidak dapat menentukan apakah menjalankan method max (int,double) ataukah max(double, int)

```
1  public class AmbiguousOverloading {
2      public static void main(String[] args) {
3          System.out.println(max(1, 2));
4      }
5
6      public static double max(int num1, double num2){
7          if (num1 > num2)
8              return num1;
9          else
10             return num2;
11     }
12
13     public static double max(double num1, int num2){
14         if (num1 > num2)
15             return num1;
16         else
17             return num2;
18
19         }
20 }
```

Variable Scope

Scope variabel adalah bagian program dimana variabel dapat diakses dan digunakan. Apabila sebuah variable didiefinisikan didalam sebuah method maka variabel tersebut disebut sebagai variabel lokal. Variabel yang dikenal disemua method disebut dengan variabel global.

Scope (jangkauan) variabel dimulai dari saat dideklarasikan sampai dengan akhir dari blok tersebut. Dari gambar di bawah ini terlihat bahwa scope variabel i dimulai dari saat dideklarasikan sampai dengan akhir dari loop for tersebut. Scope variabel j dimulai dari saat dideklarasikan sampai dengan akhir dari loop for. Variabel j tidak dapat digunakan ataupun diakses sebelum variabel tersebut dideklarasikan



Sumber : Introduction to Java Programming, Y. Daniel . 2012 (p.196)

Perhatikan contoh di bawah ini.

```
1  public class VarLocalGlobal {
2
3      public static void main(String[] args){
4          int i=10,x=20;
5          System.out.println("Nilai variabel i di main program : "+i);
6          System.out.println("Nilai variabel x di main progoram : "+x);
7          test(i);
8          System.out.println("Nilai variabel i di main program : "+i);
9          System.out.println("Nilai variabel x di main progoram : "+x);
10     }
11
12     public static void test(int i) {
13         i++;
14         int x=10;
15         System.out.println("Nilai variabel i di method test : "+i);
16         System.out.println("Nilai variabel x di method test : "+x);
17     }
18 }
```

```
Nilai variabel i di main program : 10
Nilai variabel x di main progoram : 20
Nilai variabel i di method test : 11
Nilai variabel x di method test : 10
Nilai variabel i di main program : 10
Nilai variabel x di main progoram : 20
```

- Pada program di atas terdapat dua variabel yang sama yaitu i dan x . i dan x yang dideklarasikan pada main, hanya di kenal di main, sehingga apabila nilai i di main ingin digunakan pada method test harus dilakukan passing parameter by value.
- Demikian juga dengan nilai i dan x yang ada di method test, hanya dikenal di method test dan tidak dikena di main. Sehingga saat dilakukan increment pada variabel i dan assignment nilai 10 ke variabel x, hal ini tidak mengubah nilai variabel i dan x yang ada di void main.

SIMPULAN

Pada pemrograman lebih lanjut perlu mempelajari tentang efisiensi dalam pembuatan program dalam hal ini dapat dicapai dengan menggunakan ***method***. Method yang merupakan kumpulan baris proses membantu mengurangi penggunaan kode yang sama berkali-kali dan juga membantu dalam pembentukan struktur program yang lebih rapi.

Pendefinisian method dengan nama yang sama dimungkinkan dan hal ini dikenal dengan nama overloading method. Adapun yang harus menjadi perhatian adalah method signature yang harus berbeda diantara method dengan nama sama tersebut.

Selain method yang juga harus diperhatikan adalah scope dari variabel. Dimana scope variabel akan dimulai dari saat dideklarasikan sampai dengan di akhir blok tersebut.

DAFTAR PUSTAKA

1. Y. Daniel Liang. (2012). Introduction to Java Programming :Comprehensive Version. International Edition. 09. Pearson Education. New Jersey.
2. http://java.sun.com/docs/books/tutorial/java/javaOO/arguments.html
3. http://java.sun.com/docs/books/tutorial/java/javaOO/methods.html
4. http://www.otherwise.com/Lessons/MethodsCreationAndUse.html
5. https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html
6. https://docs.oracle.com/javase/tutorial/java/javaOO/arguments.html